Bachelor-Thesis

Entwicklung einer SharePoint App zum Datenabgleich zwischen Zoho CRM und JIRA

Marcel Schmid
Im kleinen Grünle 1
77716 Haslach im Kinzigtal

Hochschule Offenburg
Elektrotechnik und Informationstechnik
Angewandte Informatik

Betreuer Hochschule: Prof. Dr. Hartwig Grabowski
Betreuer GEOCEPT: Dipl.-Inf. (FH) Michael Ring

Abstract

Diese Arbeit befasst sich mit der Konzeption und Entwicklung einer SharePoint App, die einen Datenabgleich zwischen Zoho CRM und JIRA ermöglicht. Mittels dieser App ist es möglich, einen Supportfall, den der Support mittels Zoho CRM aufgenommen hat, bei Bedarf der Entwicklung zuzuweisen, indem in JIRA ein neues Ticket erstellt wird. Des Weiteren können durch die App Informationen, wie Firmen- und Kontaktdaten beiden Abteilungen zugänglich gemacht werden. Realisiert wird dies auf Basis von SharePoint, auf das alle beteiligten Mitarbeiter Zugriff haben. Zunächst wird für eine Lösung mit SharePoint dessen App-Modell analysiert und die möglichen Hosting-Varianten evaluiert. Die dabei getroffene Entscheidung für eine SharePoint gehostete App wird zudem begründet. Um die Bearbeitung eines Supportfalls innerhalb der App zu ermöglichen, wird ein Workflow entworfen, der neben dem Ablauf der Bearbeitung auch die Kommunikation mit den beteiligten Anwendungen festlegt. Die zu diesem Zweck notwendige persistente Speicherung der Daten in der App erfolgt dabei mithilfe von SharePoint Listen. Diese werden aufgrund eines ähnlichen Aufbaus als Tabellen dargestellt. Die detaillierte Darstellung einzelner Listenelemente dagegen erfolgt mithilfe von Dialog-Fenstern, die von SharePoint bereitgestellt werden. Dabei werden abhängig von dem Land aus dem die App aufgerufen wird, die Texte auf Deutsch oder auf Englisch angezeigt.

Vorwort

An dieser Stelle möchte ich allen danken, die mich während meines Studiums und besonders während der Bachelor-Thesis unterstützt haben. Besonderer Dank gilt meinen Betreuern Prof. Dr. Hartwig Grabowski und Dipl.-Inf. (FH) Michael Ring, die mich stets tatkräftig unterstützt und beraten haben. Außerdem möchte ich allen Mitarbeitern der Firma GEOCEPT GmbH danken, an die ich mich bei Fragen jederzeit wenden konnte und die für ein tolles Arbeitsklima gesorgt haben.

Urheberrecht

Eidesstattliche Erklärung

Hiermit versichere ich eidesstattlich, dass die vorliegende Bachelor-Thesis von mir selbstständig und ohne unerlaubte fremde Hilfe angefertigt worden ist, insbesondere, dass ich alle Stellen, die wörtlich oder annähernd wörtlich oder dem Gedanken nach aus Veröffentlichungen, unveröffentlichten Unterlagen und Gesprächen entnommen worden sind, als solche an den entsprechenden Stellen innerhalb der Arbeit durch Zitate kenntlich gemacht habe, wobei in den Zitaten jeweils der Umfang der entnommenen Originalzitate kenntlich gemacht wurde. Ich bin mir bewusst, dass eine falsche Versicherung rechtliche Folgen haben wird.

....

Datum Unterschrift

Inhaltsverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
CAML	Collaborative Application Markup Language
CRM	Customer-Relationship-Management
CSOM	Client-side Object Model
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JSOM	JavaScript Object Model
JSON	JavaScript Object Notation
REST	Representational State Transfer
SOP	Same Origin Policy
SQL	Structured Query Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	Extensible Markup Language

Abbildungsverzeichnis

Tabellenverzeichnis

Listing-Verzeichnis

1. Einleitung

In diesem Abschnitt wird zu Beginn die Motivation beschrieben, die zu dieser Arbeit führte. Dabei werden sowohl die Ausgangssituation als auch die zugrunde liegende Problemstellung genauer betrachtet. Abschließend wird aufgrund dessen die Zielsetzung dieser Arbeit beschrieben.

1.1. Motivation

Die GEOCEPT GmbH entwickelt Telematik-Software für viele Unternehmen, die in unterschiedlichen Branchen tätig sind. Entsprechend wichtig ist die Verwaltung von Firmen- und Kontaktdaten vor allem im Hinblick auf den Support. Eingehende Supportanfragen müssen den entsprechenden Kunden, auch im Nachhinein noch, zugeordnet werden können. Der Einsatz von Anwendungen in diesem Bereich ist deshalb nahezu unverzichtbar. Zukünftig soll bei GEOCEPT für diese Aufgaben Zoho CRM eingesetzt werden. Damit lassen sich unter anderem Firmen und Kontakte verwalten und Supportfälle anlegen, zu denen angegeben werden kann, welche Firma beziehungsweise welcher Kontakt ihn gemeldet hat.

Nicht selten besteht die Anforderung bestimmte Supportfälle der Entwicklungsabteilung zuzuweisen, damit sie aufgetretene Probleme analysieren und beheben können. Für die Verwaltung und Dokumentation der anstehenden Aufgaben wird hier allerdings JIRA eingesetzt. Dabei handelt es sich um eine Anwendung, mit der eine Aufgabe angelegt, einer Person zur Bearbeitung zugewiesen und der Arbeitsfortschritt dokumentiert werden kann.

Standardmäßig gibt es keine Kommunikation, geschweige denn einen Datenaustausch, zwischen den genannten Anwendungen. Es wäre jedoch wünschenswert, dass ein Supportmitarbeiter den Fortschritt der Bearbeitung eines Supportfalls einsehen kann, ohne den Entwickler fortwährend danach fragen zu müssen. Andersherum kann es auch sinnvoll sein, dass ein Entwickler die Firmen- und Kontaktinformationen der Firma einsehen kann. Um dieses Problem zu lösen, müsste allen Mitarbeitern der Zugriff auf beide Anwendungen gewährt werden. Dies ist allerdings mit hohen Kosten verbunden. Die Hersteller der Anwendungen verlangen für die Nutzung Lizenzen, mit denen der Zugriff ermöglicht wird. Werden mehr Lizenzen benötigt, steigen die Kosten somit um ein Vielfaches an.

Im Rahmen dieser Arbeit soll aus diesem Grund eine Anwendung entwickelt werden, die den Datenaustausch zwischen Zoho CRM und JIRA regelt. Dies soll auf Basis von SharePoint geschehen, das bei GEOCEPT bereits zur zentralen Verwaltung von Informationen und Daten eingesetzt

wird. Die zu entwickelnde Anwendung soll deshalb als SharePoint App realisiert werden, die in der vorhandenen SharePoint Instanz installiert und verwendet werden kann. Diese App soll die Daten der beiden Anwendungen abfragen und entsprechend aufbereiten, sodass keine zusätzlichen Lizenzen benötigt werden. Jeder der Mitarbeiter hat bereits Zugriff auf SharePoint, dadurch kann er die App nutzen und bekommt somit indirekt Zugriff auf die Daten von Zoho CRM und JIRA. Die Abbildung 1.1 veranschaulicht den beschriebenen Lösungsansatz.

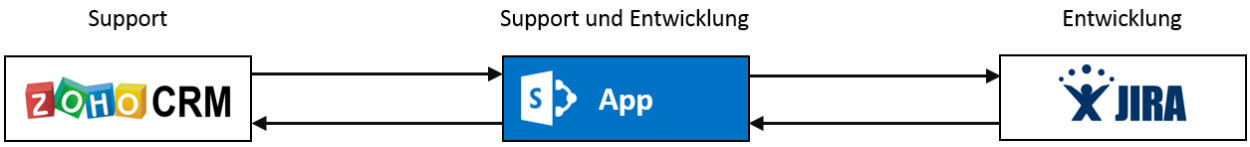

Abbildung 1.1.: Übersicht des Lösungsansatzes

1.2. Zielsetzung

Die Zielsetzung dieser Arbeit ist die Konzeption und Entwicklung einer SharePoint App. Diese soll keine eigenständige Ticketing- oder CRM-Anwedung werden, sondern den Datenabgleich zwischen Zoho CRM und JIRA regeln. Eine Grundvoraussetzung dafür ist es, das in SharePoint neu eingeführte App-Modell zu analysieren und die für die Problemstellung passende Variante zu wählen. Ein weiterer zentraler Aspekt ist es, eine Möglichkeit zu finden, wie Daten in der App persistent gespeichert werden können. Darüber hinaus ist eine geeignete Darstellung dieser Daten zu wählen, sowie ein Workflow zu entwickeln, der die Bearbeitung eines Supportfalls innerhalb der App beschreibt. Dieser Workflow muss unter andrem regeln, wie Zoho CRM und JIRA in der Verarbeitung eines Supportfalls eingebunden werden können und muss die dazu notwendige Kommunikation festlegen.

2. Grundlagen

In diesem Abschnitt wird zuerst kurz die Firma GEOCEPT GmbH, sowie die Anwendungen SharePoint, Zoho CRM und JIRA vorgestellt. Anschließend werden die in dieser Arbeit eingesetzten Technologien beschrieben, bevor abschließend der Stand der Technik in Bezug auf die Aufgabenstellung betrachtet wird.

2.1. GEOCEPT GmbH

Die GEOCEPT GmbH ist ein Unternehmen, das Telematik-Software für unterschiedliche Branchen entwickelt. Bei Telematik handelt es sich um eine Technik, die Telekommunikation und Informatik verknüpft und zu dessen Kernbereichen unter anderem Verkehrstelematik und Flottenmanagement zählen [1]. Zu diesen Bereichen bietet die GEOCEPT GmbH eigenentwickelte Software an, die eine effiziente Routenplanung ermöglicht und flexibel auf Änderungen reagiert. Dabei können die Routen der Fahrzeuge auch im Büro verfolgt und bei Auftragsänderung gegebenenfalls angepasst werden. Zum Einsatz kommt diese Software beispielsweise bei Entsorgungs-, Geld- und Werttransportunternehmen [2].

2.2. SharePoint

SharePoint ist eine von Microsoft entwickelte Webanwendung, die eine Vielzahl verschiedener Einsatzmöglichkeiten bietet. So umfasst SharePoint unter anderem die Bereiche Zusammenarbeit, soziale Netzwerke, Intranet Portale, Content-Management und Geschäftsanwendungen [3]. Die Arbeit innerhalb einer Abteilung oder eines Unternehmens soll dadurch in möglichst vielen Bereichen erleichtert werden. Durch das in SharePoint 2013 neu eingeführte App-Modell können Entwickler SharePoint-Erweiterungen in Form von „Apps“ bereitstellen. Diese können über den offiziellen Office Store vertrieben und bezogen werden, sodass die Funktionalität von SharePoint durch Eigenentwicklungen von Drittanbietern erweitert werden kann.

Bei einer SharePoint App handelt es sich nicht, wie vermutet werden könnte, um eine mobile Anwendung, sondern um eine Möglichkeit den Funktionsumfang von SharePoint durch eine kompakte Anwendung zu erweitern. Dabei ist eine SharePoint App ein einzelnes Paket, das alle Seiten, Skripte, Styles und SharePoint Komponenten enthält, die benötigt werden, um die Anwendung ausführen zu können [4]. Somit ist eine einfache Installation und restlose Deinstallation

möglich. Für die Bereitstellung dieser Apps gibt es grundsätzlich zwei Varianten.

- **Von Anbieter gehostet:** Hierbei werden die Daten und die Geschäftslogik auf externen Servern bereitgestellt. Diese Apps können zusätzlich auch SharePoint Komponenten enthalten, welche direkt dort gehostet werden. Dies ermöglicht es den Entwicklern sowohl SharePoint, als auch externe Komponenten in der App zu verwenden.
- **In SharePoint gehostet:** Die gesamten Daten werden direkt in SharePoint bereitgestellt. Somit können bei dieser Variante ausschließlich die SharePoint Komponenten genutzt werden. Es entfällt dafür die Notwendigkeit eines externen Servers.

Unabhängig von der Bereitstellung können in einer App SharePoint Komponenten genutzt werden. Dabei handelt es sich beispielsweise um Seiten oder Listen, die in den meisten Apps zum Einsatz kommen.

2.3. Zoho CRM

Der Begriff „Customer-Relationship-Management“ (kurz CRM) beschreibt sowohl die Unternehmensstrategie den Fokus verstärkt auf Kunden zu richten, als auch die Anwendungen, die das Unternehmen bei diesem Vorhaben unterstützen [5]. Zoho CRM ist solch eine Anwendung. Es können damit Firmen- und Kontaktdaten verwaltet werden, wobei die jeweiligen Felder individuell an die eigenen Gegebenheiten angepasst werden können. Zudem können Supportfälle erstellt und mit den Firmen beziehungsweise Kontakten verknüpft werden, die sie gemeldet haben. Generell können mit Zoho CRM sämtliche Informationen über den gesamten Prozess der Kundenakquise gespeichert und ausgewertet werden. Auch eine grafische Aufbereitung dieser Daten wird von der webbasierten CRM-Anwendung unterstützt.

Zoho stellt für ihre CRM-Anwendung eine eigene API bereit, damit Fremdanwendungen mit den Daten von Zoho CRM interagieren können. Es handelt sich dabei um eine REST (siehe Abschnitt 2.5.2) API, die die Daten wahlweise als XML (siehe Abschnitt 2.5.4) oder JSON (siehe Abschnitt 2.5.3) formatiert und somit programmiersprachenunabhängig zurück liefert. Die Anfragen sind auf eine bestimmte Anzahl pro Tag beschränkt, abhängig davon, welche Version von Zoho CRM eingesetzt wird. Auch die Anzahl der Einträge pro Anfrage sind begrenzt. Um sich mit der API zu authentifizieren, wird ein Authentifizierungstoken benötigt. Dieser kann zu einer Kombination aus Benutzername und Passwort generiert werden. Damit die Anfragen akzeptiert werden, muss der Benutzer, zu dem der Token gehört, über die entsprechenden Rechte verfügen.

2.4. JIRA

„JIRA ist eine webbasierte Anwendung zur Fehlerverwaltung, Problembehandlung und operativen Projektmanagement“ [6]. JIRA wird in vielen verschiedenen Bereichen eingesetzt, wird allerdings

am häufigsten bei der Softwareentwicklung genutzt. Unter anderem unterstützt es dort das Anforderungsmanagement. Für all diese Aufgaben können in JIRA Tickets angelegt werden. Jedes dieser Tickets wird einem Projekt zugeordnet und enthält alle für die Bearbeitung benötigten Informationen. Während der Bearbeitung können Kommentare und Dateien angehängt werden, womit auch eine Dokumentation erreicht wird. Zudem ist es möglich Tickets, auch in Abhängigkeit des Projekts, anzupassen und beispielsweise eigene Felder hinzuzufügen. Die Bearbeitung des Tickets durchläuft verschiedene Status, die durch einen anpassbaren Workflow definiert sind. Durch die Auswertung des Status innerhalb eines Projekts kann dessen Fortschritt erfasst werden.

Auch JIRA stellt für Fremdanwendungen eine REST (siehe Abschnitt 2.5.2) API bereit. Diese erlaubt die Interaktion mit den Daten von JIRA, wobei der gesamte Datenaustausch mittels JSON (siehe Abschnitt 2.5.3) erfolgt. Im Gegensatz zu der Zoho CRM API gibt es hier keine Beschränkungen der Anfragen. Außerdem stehen hier mehrere Möglichkeiten zur Authentifizierung zur Verfügung. Allerdings muss auch hier der Benutzer über die entsprechenden Rechte verfügen, damit die Anfragen erfolgreich ausgeführt werden können.

2.5. Eingesetzte Technologien

In diesem Abschnitt werden die wichtigsten Technologien beschrieben, die für die Umsetzung dieser Arbeit eingesetzt wurden.

2.5.1. jQuery

Bei jQuery handelt es sich um eine freie Bibliothek für JavaScript. Es ist die am häufigsten verwendete JavaScript-Bibliothek und wird auf jeder zweiten Webseite verwendet [7]. Es bietet viele Funktionalitäten, die den Umgang mit JavaScript erleichtern oder erweitern [8], [9]. Die wichtigsten Funktionen sind dabei:

- einfache Selektion von Elementen
- DOM (Document Object Model) Manipulation und Navigation
- erweitertes Event-System
- Effekte und Animationen
- diverse Hilfsfunktionen
- Erweiterbarkeit durch zahlreiche Plug-ins

Zudem kann jQuery in den meisten Browser eingesetzt werden. Auch wenn diese sich unter anderem darin unterscheiden, wie sie JavaScript-Code interpretieren. So kann derselbe Code bei den

Browsern zu unterschiedlichen Verhalten führen. Die jQuery-Funktionen dagegen funktionieren trotz dieser Inkompatibilitäten auf allen Browsern gleich.

Ein weiterer Vorteil, den jQuery bietet, ist das HTML- und JavaScript-Code vollständig voneinander getrennt werden können. Statt in den HTML-Attributen JavaScript-Funktionen aufzurufen, kann dasselbe Verhalten mit jQuery rein mit JavaScript-Mitteln erreicht werden. Dies ist vor allem dadurch möglich, dass jQuery die Möglichkeit bietet HTML-Elemente, mit den aus CSS bekannten Selektoren, zu selektieren und ihnen beispielsweise entsprechende Events hinzuzufügen oder sie anderweitig zu manipulieren. In Listing 2.1 wird dies veranschaulicht.

```
1 $(document).ready(function () {
2     $('#btn').click(function () {
3         // Wird bei jedem Klick auf den Button ausgeführt!
4     });
5 });
```

Listing 2.1: Hinzufügen eines Events mit jQuery

Wie im Listing 2.1 zu erkennen ist, wird das HTML-Element mit der ID *btn* selektiert und diesem ein Klick-Event hinzugefügt. Dies geschieht innerhalb der *ready* Funktion, die den typischen Einstiegspunk für jQuery darstellt. Diese wird ausgeführt, nachdem der DOM-Baum aufgebaut wurde.

2.5.2. REST

Bei REST handelt es sich um einen Architekturstil für Webanwendungen. Es gibt keinen Standard, der vorschreibt, wie eine REST-Anwendung auszusehen hat, dennoch gibt es einige Randbedingungen [10].

- **Client-Server:** Durch das Trennen der Angelegenheiten der Clients von denen des Servers können die Clients einfach auf andere Plattformen portiert werden.
- **Zustandslos:** Die Kommunikation verläuft ausschließlich zustandslos. Der Server hat demnach kein Vorwissen über die Clients oder über vorangegangene Anfragen.
- **Cache:** Die Antworten können von Clients im Cache gehalten werden, dazu werden sie vom Server entsprechend als „cacheable“ gekennzeichnet.
- **Einheitliche Schnittstellen:** Die Architektur soll durch die einheitlichen Schnittstellen vereinfacht und entkoppelt werden. So wird beispielsweise jede Ressource durch eine URI eindeutig identifiziert.
- **Mehrschichtiges System:** Es kann beliebig viele Stationen zwischen Client und Server geben, wobei der Client nicht weiß, ob er direkt mit dem Endserver verbunden ist oder nicht.

- **Code bei Bedarf:** Der Server kann die Funktionalität des Clients erweitern, indem er ausführbaren Code verschickt. Dies kann jedoch von den Clients aus Sicherheitsgründen unterbunden werden. Aus diesem Grund handelt es sich hierbei um eine optionale Bedingung.

Web-Services, die sich an diese Randbedingungen halten, können als „RESTful“ bezeichnet werden [11]. Verletzt ein Service eine der Bedingungen, kann er dagegen nicht mehr als „RESTful“ betrachtet werden.

Obwohl REST für sich kein Standard ist, nutzen die meisten RESTful Webservices Standards, wie beispielsweise HTTP, URI, JSON (siehe Abschnitt 2.5.3) oder XML (siehe Abschnitt 2.5.4) [12]. So baut REST auf HTTP auf und nutzt dessen Operationen. Mit Hilfe von URIs werden einzelne Ressourcen eindeutig identifiziert und JSON beziehungsweise XML werden häufig als Datenaustauschformat eingesetzt.

Die HTTP Operationen werden typischerweise wie in der Tabelle 2.1 dargestellt verwendet. Jedoch handelt es sich hierbei nur um Richtlinien, die nicht zwingend eingehalten werden müssen.

Tabelle 2.1.: Verwendung der HTTP-Operation bei REST (nach [12])

Ressource	GET	PUT	POST	DELETE
URI einer Collection, zum Beispiel: *http://beispiel.de/Liste/*	Liefert die Collection zurück	Ersetzt die Collection mit einer anderen	Fügt der Collection ein neues Element hinzu und liefert es zurück	Löscht die Collection
URI eines Elements, zum Beispiel: *http://beispiel.de/Liste/Element*	Liefert das Element zurück	Ersetzt das Element oder erstellt ein neues, wenn es nicht existiert	-	Löscht das Element

2.5.3. JSON

JSON ist ein kompaktes Datenaustauschformat. Es stellt in diesem Bereich eine Alternative zu XML (siehe Abschnitt 2.5.4) dar, wobei es im Gegensatz dazu weniger Ressourcen (wie beispielsweise Speicherplatz) benötigt. Es lehnt sich stark an die JavaScript-Objektdefinition an, ist jedoch völlig programmiersprachenunabhängig. JSON bietet den Vorteil, dass es von Menschen leicht gelesen und geschrieben und von Maschinen einfach generiert und übersetzt werden kann [13]. Dies ist einer der Gründe, warum es vermehrt eingesetzt wird und es für jede gängige Programmiersprache entsprechende Parser gibt. Vor allem im Umfeld von Web-Services findet JSON vermehrt Einsatz um Daten auszutauschen.

Die Hauptkomponenten aus denen JSON besteht sind Objekte, Sammlungen von Name-Wert Paaren, Arrays und Listen von Werten. Als Werte können Zeichenketten, Zahlen oder wiederum

Objekte beziehungsweise Arrays verwendet werden. So ist es möglich, beliebig komplexe Strukturen aufzubauen.

2.5.4. XML

XML ist eine „Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien“ [14], die sowohl menschen-, als auch maschinenlesbar ist. Es bietet viele Einsatzmöglichkeiten, wobei der plattform- und programmiersprachenunabhängige Austausch von Daten einer der Hauptverwendungszwecke darstellt.

Der Aufbau einer XML-Datei kann durch eine Schemasprache vorgegeben werden, wodurch es möglich ist, eine XML-Datei auf ihre Gültigkeit, in Bezug auf den Aufbau, zu prüfen. Aufgrund der weiten Verbreitung gibt es für zahlreiche Programmiersprachen entsprechende Parser, die teilweise auch die Prüfung der Gültigkeit, sowohl in Bezug auf den Aufbau, als auch auf die Wohlgeformtheit, übernehmen. „Wohlgeformt“ bedeutet, dass eine XML-Datei alle XML-Regeln einhält. Darunter fällt beispielsweise, dass ein XML-Dokument nur einen Wurzelknoten besitzen darf.

2.6. Stand der Technik

In diesem Abschnitt werden Lösungsmöglichkeiten für eine Integration von CRM und Issue-Tracker Anwendungen betrachtet. Dabei sollen sowohl die Möglichkeiten zur Lösung als auch die Funktionalität der Anwendungen aufgezeigt werden.

2.6.1. Zoho CRM plugin for JIRA

Die Funktionalität von JIRA kann mittels Plug-ins erweitert werden. Diese stammen von Drittanbietern und werden über den Atlassian Marketplace vertrieben. „Zoho CRM plugin for JIRA“ ist ein solches Plug-in und versucht darüber Zoho CRM in JIRA zu integrieren.

Mit diesem Plug-in können zu einem Ticket unter anderem Firmen- oder Kontaktdaten aus Zoho CRM dargestellt werden. Dazu gibt es beim Erstellen eines JIRA Tickets zusätzliche Felder, in die entsprechende Informationen aus Zoho CRM eingetragen werden. Mit deren Hilfe holt das Plug-in den entsprechenden Eintrag von Zoho CRM. Soll mehr als ein Zoho CRM Eintrag einem Ticket hinzugefügt werden, kann eine eigene Suche definiert werden. Mit dieser wird festgelegt, wie in Zoho CRM nach Einträgen gesucht werden soll. Alle Einträge, die dem Suchkriterium entsprechen, werden so im JIRA Ticket angezeigt.

Dieses Plug-in ermöglicht lediglich eine einseitige Integration von Zoho CRM in JIRA. So ist es nicht möglich Information von JIRA in Zoho CRM anzuzeigen oder von Zoho CRM aus ein JIRA Ticket anzulegen. Außerdem kann der Anwender nicht festlegen, welche Informationen zu einem Eintrag angezeigt werden sollen.

2.6.2. Zapier

Zapier ist eine Webanwendung, die verschiedene Anwendungen mit Hilfe von sogenannten „Zaps“ miteinander verbindet. „Ein Zap ist eine Verbindung zwischen zwei Anwendungen bestehend aus einem Event und einer Aktion“ [15]. Jedes Mal, wenn das Event auftritt, führt Zapier automatisch die damit verbundene Aktion aus. Es werden für viele Anwendungen entsprechende Events und Aktionen bereitgestellt, so auch für JIRA und Zoho CRM.

Um mit Zapier eine Integration von JIRA und Zoho CRM zu erreichen, können beliebig viele Zaps aus beliebigen Events und Aktionen angelegt werden. Dazu stehen für JIRA und Zoho CRM Events und Aktionen bereit, auf die reagiert beziehungsweise die ausgeführt werden können. Dies ermöglicht eine einfache, an die Bedürfnisse des Anwenders angepasste, Integration der beiden Anwendungen.

Auch wenn mit Zapier eine Integration von JIRA und Zoho CRM möglich ist, sind die Möglichkeiten beschränkt, da nur die von Zapier vorgegebenen Events und Aktionen verwendet werden können um einen Zap anzulegen. So können beispielsweise keine Daten von Zoho CRM in JIRA dargestellt werden, ohne das dabei ein neues Ticket angelegt wird. Ebenso ist es nicht möglich in Zoho CRM Informationen von JIRA anzuzeigen, ohne einen neuen Eintrag zu erstellen. Außerdem werden die Zaps jedes Mal ausgeführt, sobald das Event eintritt. Somit gibt es keine Möglichkeit zu steuern, dass die Aktion nur in bestimmten Fällen ausgeführt werden soll.

3. Konzeption

Dieser Abschnitt befasst sich mit der Konzeption der SharePoint App, die im Rahmen dieser Arbeit entwickelt wurde. Dazu werden zu Beginn die Voraussetzungen bei der GEOCEPT GmbH beschrieben, die bei der Entwicklung der App zu beachten waren. Anschließend werden die Anforderungen an die App genannt, die es umzusetzen galt. Danach folgt die Evaluation der Hosting-Varianten. Dabei wird neben den verschiedenen Varianten auch die getroffene Entscheidung beschrieben. Abschließend wird der Aufbau und die Kommunikation der App mit den REST APIs von Zoho CRM und JIRA genauer betrachtet.

3.1. Voraussetzungen

In der GEOCEPT GmbH soll im Support zukünftig Zoho CRM eingesetzt werden, um Firmen- und Kontaktdaten zu verwalten sowie Supportfälle zu erfassen. In der Entwicklung wird bereits JIRA eingesetzt, mit dem Aufgaben erfasst und deren Bearbeitung dokumentiert wird. Im Rahmen dieser Arbeit soll ein Datenaustausch zwischen diesen beiden Anwendungen entwickelt werden. Zu Supportfällen, die in Zoho CRM vom First-Level-Support erstellt wurden, soll es möglich sein, Tickets in JIRA zu generieren und der Entwicklung zur Bearbeitung zuzuweisen.

Als Bindeglied zwischen den beiden Anwendungen soll SharePoint eingesetzt werden. Dies hat mehrere Gründe. Ein Grund ist, dass im Rahmen der vorhandenen Office 365 Lizenz für jeden Mitarbeiter eine SharePoint-Lizenz zur Verfügung steht. Darüber hinaus ist für die Entwicklung und Verwendung von SharePoint-Lösungen keine spezielle Infrastruktur notwendig. Es werden keine weiteren Server, beispielsweise für die Datenbank, benötigt, da SharePoint auf einem oder mehreren Servern aufsetzt und diese selbstständig verwaltet. SharePoint bietet dem Programmierer zudem ein fertiges Set aus Werkzeugen, um Businesslogik effizient erstellen zu können.

Bei der in Office 365 enthaltenen SharePoint-Lizenz handelt es sich um die 2013er Version, die das neue App-Modell (siehe Abschnitt 2.2) nutzt. Aus diesem Grund soll die zu entwickelnde Lösung als SharePoint App umgesetzt werden. Allerdings bietet die genutzte SharePoint-Lizenz im Rahmen von Office 365 nur einen eingeschränkten Funktionsumfang. Dies hat zur Folge, dass mit dieser keine eigenen Apps entwickelt oder verwendet werden können. Nichtsdestotrotz ist es möglich, den Funktionsumfang durch das Installieren von Apps aus dem offiziellen Office Store zu erweitern. Ziel ist es demnach die zu entwickelnde App in erster Linie für GEOCEPT interne Zwecke im Office Store zu veröffentlichen. Dies ist der einzige Weg um die eigene App auf den

Office 365 SharePoint Server verwenden zu können.

Damit die App entwickelt und getestet werden kann, muss eigens dafür eine zusätzliche SharePoint Instanz auf einem GEOCEPT hausinternen Testserver bereitgestellt werden. Im Rahmen dieser Arbeit wurde eine solche Instanz für die Entwicklung der App eingesetzt.

3.2. Anforderungen

Im Zuge der Konzeption wurden Anforderungen an die SharePoint App aufgestellt. Dabei wurden neben funktionale auch allgemeine Anforderungen definiert, die die App erfüllen sollte. In diesem Abschnitt werden diese Anforderungen aufgelistet und beschrieben.

3.2.1. Allgemeine Anforderungen

Bei diesen Anforderungen handelt es sich um Rahmenbedingungen für die App. Diese zielen auf die notwendige Veröffentlichung der SharePoint App ab.

- **Veröffentlichung im Office Store**
  Die App muss dazu die Anforderungen von Microsoft für die Veröffentlichung erfüllen. Dazu gehört unter anderem, dass die App stabil und funktionsfähig sein muss.
- **Sprachunterstützung von Deutsch und Englisch**
  Um eine größere Zielgruppe anzusprechen, soll die App nicht nur Deutsch, sondern auch Englisch als Sprache unterstützen.
- **Verwendung ohne JIRA und oder Zoho CRM möglich**
  Dies soll gewährleisten, dass die App auch verwendet werden kann, wenn nur eine oder keine der beiden Anwendungen zur Verfügung steht. Allerdings mit der Einschränkung, dass nicht alle Funktionen, die JIRA oder Zoho CRM erfordern genutzt werden können.

3.2.2. Funktionale Anforderungen

Nachfolgend werden alle funktionalen Anforderungen an die App beschrieben. Dabei handelt es sich um die wichtigsten Funktionalitäten, die somit den Aufbau der App festlegen.

- **Daten zur Authentifizierung speichern**
  Die Informationen für die Authentifizierung bei JIRA beziehungsweise Zoho CRM müssen in der App gespeichert werden können, um der App den Zugriff auf die jeweilige Anwendung zu ermöglichen. Dazu zählt auch die URI zu der jeweiligen Anwendung, da diese sich bei verschiedenen Anwendern unterscheidet. Zudem soll es möglich sein diese Informationen zu bearbeiten oder bei Bedarf zu löschen.

- **Abruf der Daten von JIRA und Zoho CRM**
  Für das Abrufen der Daten von Zoho CRM und JIRA soll die App Buttons bereitstellen, mit denen die jeweilige Aktion ausgeführt werden kann. Ruft die App Daten von Zoho CRM ab, werden die bereits vorhandenen Supportfälle aktualisiert und die neuen der App hinzugefügt. Bei dem Abruf der Daten von JIRA werden alle von der App aus angelegten Tickets mit JIRA abgeglichen und falls nötig aktualisiert.

- **Synchronisierung mit Zoho CRM**
  Eine Synchronisation der in der App gespeicherten Supportfälle zurück nach Zoho CRM ist ebenfalls zu ermöglichen. Dies soll per Knopfdruck für alle Supportfälle oder beim Schließen eines Supportfalls geschehen.

- **Auflisten und filtern von Supportfällen**
  In der App können alle Supportfälle aufgelistet und bei Bedarf nach ihrem Status gefiltert werden.

- **Bearbeitung von Supportfällen**
  Jeder Supportfall kann in der App bearbeitet werden. Dabei können verschiedene Informationen aktualisiert werden, darunter auch der Status nach dem im Abschnitt 3.4.3 beschriebenen Workflow.

- **Supportfälle der Entwicklung zuweisen**
  Durch die Zuweisung eines Supportfalls zu der Entwicklung erstellt die App ein neues Ticket in JIRA und speichert dessen Informationen zu dem Supportfall.

- **Firmen- und Kontaktliste darstellen**
  In der App können die Firmen und Kontakte von Zoho CRM dargestellt werden. Dabei handelt es sich um einen rein lesenden Vorgang, das bedeutet eine Bearbeitung in der App ist nicht notwendig.

3.3. Evaluation der Hosting-Varianten

Für das von Microsoft in SharePoint 2013 eingeführte App-Modell gibt es verschiedene Varianten, die sich in der Art und Weise wie die App bereitgestellt wird, unterscheiden. Für die zu entwickelnde App bedeutete dies, dass zunächst die für das Problem passende Variante gewählt werden musste. In den nachfolgenden Abschnitten werden deshalb die beiden Varianten genauer betrachtet, bevor im letzten Abschnitt die getroffene Wahl begründet wird.

3.3.1. Anbieter gehostete App

Bei einer Anbieter gehosteten App werden beim Anlegen in Visual Studio zwei Projekte erzeugt. Eines der beiden Projekte ist für die Darstellung zuständig und wird direkt in SharePoint gehostet. Alle SharePoint Komponenten werden diesem Projekt hinzugefügt. Für die Entwicklung der Benutzeroberfläche wird ASP.Net verwendet, allerdings ohne die Code-behind Funktionalität, da in diesem Projekt kein serverseitiger Code ausgeführt werden kann. Die Implementierung der Logik muss somit clientseitig erfolgen, wozu JavaScript verwendet wird.

Das andere Projekt trägt „Web“ als Suffix im Namen und wird außerhalb und unabhängig von der Office 365 SharePoint Plattform gehostet. Der Anbieter, bei dem dieser Teil der App gehostet werden soll, kann frei vom Anwender gewählt werden. So kann auch ein eigener Windows Applikationsserver oder das Cloud-Hosting Windows Azure für diesen Zweck verwendet werden. Da dieses Projekt auf einem Applikationsserver läuft, enthält es den Code, der serverseitig ausgeführt werden soll. Für die Programmierung des serverseitigen Codes wird eine .NET Sprache wie beispielsweise C# eingesetzt. Zudem ist es ebenfalls möglich ASP.Net für Benutzeroberflächen zu verwenden, wobei hier auch die Code-behind Funktion genutzt werden kann.

Bei dieser Variante gibt es somit ein Projekt für client- und ein Projekt für serverseitigen Code. Die Kommunikation zwischen diesen beiden Teilprojekten erfolgt über die REST-Schnittstelle oder über das clientseitige Objekt Modell (CSOM) von SharePoint. Wie in Abbildung 3.1 schematisch dargestellt ist, kann dadurch sowohl auf bereits in SharePoint vorhandene Seiten und Dienste als auch auf das „App Web“ zugegriffen werden. Bei Letzterem handelt es sich um die Domäne des in SharePoint gehosteten Teilprojekts.

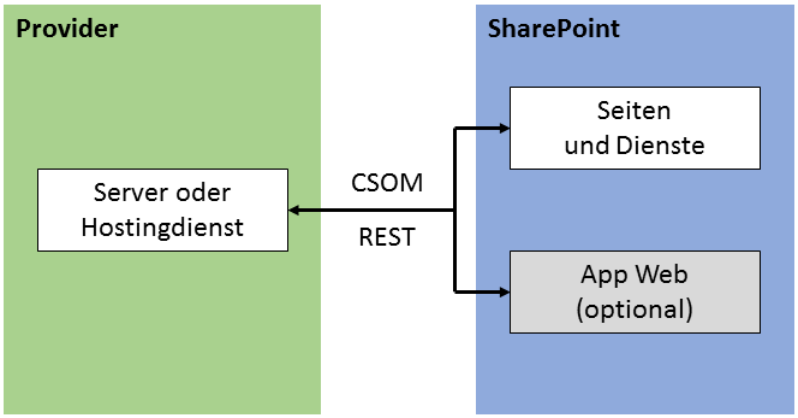

Abbildung 3.1.: Übersicht einer Anbieter gehosteten App (nach [16])

3.3.2. SharePoint gehostete App

Bei einer in SharePoint gehosteten App wird beim Anlegen in Visual Studio lediglich ein einziges Projekt erzeugt. Dieses ist identisch zu dem einen Teilprojekt bei den Anbieter gehosteten Apps, das direkt in SharePoint gehostet wird. Bei dieser Variante gibt es nur dieses Projekt, aus diesem

Grund wird dieses, im Gegensatz zu den Anbieter gehosteten Apps, nicht nur für die Benutzeroberfläche, sondern auch für die Implementierung der Logik verwendet. Es ist hierbei allerdings nicht möglich, serverseitigen Code auszuführen. Die Implementierung der gesamten Logik muss somit clientseitig erfolgen. Das bedeutet, dass die gesamte Logik mit JavaScript realisiert werden muss. Der wesentliche Vorteil ist jedoch, dass bei dieser Variante kein zusätzlicher Server oder Hostingdienst notwendig ist, um die App betreiben zu können.

Auch mit in SharePoint gehosteten Apps kann auf die bereits in SharePoint vorhandenen Seiten und Dienste, sowie auf das „App Web“ zugreifen werden, wie die Abbildung 3.2 veranschaulicht. Der Zugriff kann hier ebenfalls über die REST-Schnittstelle erfolgen oder es wird das JavaScript Objekt Modell (JSOM) von SharePoint verwendet.

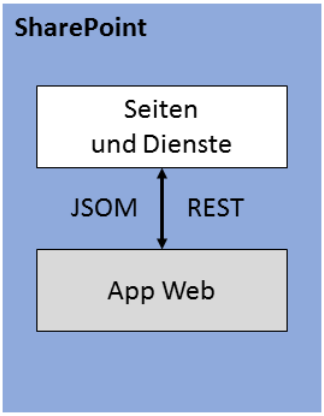

Abbildung 3.2.: Überischt einer in SharePoint gehosteten App (nach [16])

3.3.3. Entscheidung

Im Rahmen dieser Arbeit wurden die Vor- und Nachteile beider Varianten gegeneinander abgewogen und aufgrund dessen eine Entscheidung getroffen. In diesem Abschnitt werden die Vor- und Nachteile beider Varianten beschrieben, bevor anschließend die getroffene Entscheidung begründet wird.

Die Anbieter gehosteten Apps bieten zahlreiche Vorteile. So ist durch die separaten Projekte für den client- beziehungsweise serverseitigen Code eine unabhängige Entwicklung möglich. Dadurch, dass serverseitiger Code ausgeführt werden kann, können mit C# komplexe und umfangreiche Anwendungen entwickelt werden. Zudem gibt es nur wenige Beschränkungen beim Zugriff auf externe Daten oder Webservices. Nachteilig dagegen ist die aufwendigere Kommunikation zwischen den Teilprojekten, um die sich der Entwickler selbst kümmern muss. Der schwerwiegendste Nachteil ist allerdings der, dass zusätzlich ein Server oder Hostingdienst notwendig ist, um die App betreiben zu können.

Der Hauptkritikpunkt der Anbieter gehosteten Apps ist zugleich der größte Vorteil der SharePoint gehosteten Apps, den diese benötigen keinen zusätzlichen Server oder Hostingdienst. Al-

lerdings haben diese Apps andere Nachteile. So gibt es keine Möglichkeit serverseitigen Code auszuführen, was bedeutet, dass die Logik komplett clientseitig mit JavaScript implementiert werden muss. Dadurch gibt es Einschränkungen, die bei der Entwicklung beachtet werden müssen, wie beispielsweise beim Zugriff auf externe Webservices (siehe Abschnitt 4.4).

Nach der Abwägung der Vor- und Nachteile der beiden Hosting-Varianten wurde entschieden, dass im Rahmen dieser Arbeit eine SharePoint gehostete App entwickelt werden soll. Ausschlaggebend für diese Entscheidung war letztendlich, dass kein zusätzlicher Server oder Hostingdienst benötigt wird. Die Entscheidung für SharePoint als Bindeglied zwischen Zoho CRM und JIRA ist der Tatsache geschuldet, dass keine zusätzliche Serverplattform für den Betrieb der Gesamt-Infrastruktur erforderlich ist. Einzig für die App neben SharePoint einen zusätzlichen Server oder Hostingdienst einzusetzen, würde diesen Grund hinfällig machen und die Frage aufwerfen ob nicht eine andere Möglichkeit gefunden werden sollte die Aufgabe des Datenaustauschs zwischen Zoho CRM und JIRA zu lösen.

Ein wichtiger Antrieb für die Entwicklung der SharePoint App war das Ziel, der Firma GEOCEPT Kosten in Form von Benutzer-Softwarelizenzen sowohl seitens JIRA als auch bei Zoho CRM einzusparen. Einen eigenen Server bereitzustellen oder zusätzlich einen Hostingdienst in Anspruch zu nehmen würde dieses Einsparungsziel ad absurdum führen.

3.4. Aufbau der App

In diesem Abschnitt wird das grundlegende Design der App beschrieben, das im Rahmen dieser Arbeit entwickelt wurde. Dazu werden zuerst die benötigten Seiten und deren Zusammenhang dargestellt und anschließend der grobe Aufbau einer solchen Seite genauer betrachtet. Letztlich wird der Workflow der Bearbeitung eines Supportfalls und somit des Datenaustauschs zwischen Zoho CRM und JIRA beschrieben.

3.4.1. Strukturierung der Seiten

Die benötigten Webseiten für die App ergeben sich aus den in Abschnitt 3.2 beschriebenen Funktionen. Dabei soll jede Seite für einen eigenen Aufgabenbereich zuständig sein. Es gibt es für die Einstellungen sowie für die Verwaltung der Supportfälle, Firmen und Kontakte jeweils eine eigene Seite. Diese Seite enthalten Verweise zu den anderen Seiten, um den Anwender eine Navigation durch die App zu ermöglichen. Für die Bearbeitung oder die detaillierte Darstellung von Informationen werden Dialog-Fenster verwendet (siehe Abschnitt 4.3). Dadurch ergibt sich der Aufbau der App, wie er in Abbildung 3.3 dargestellt ist.

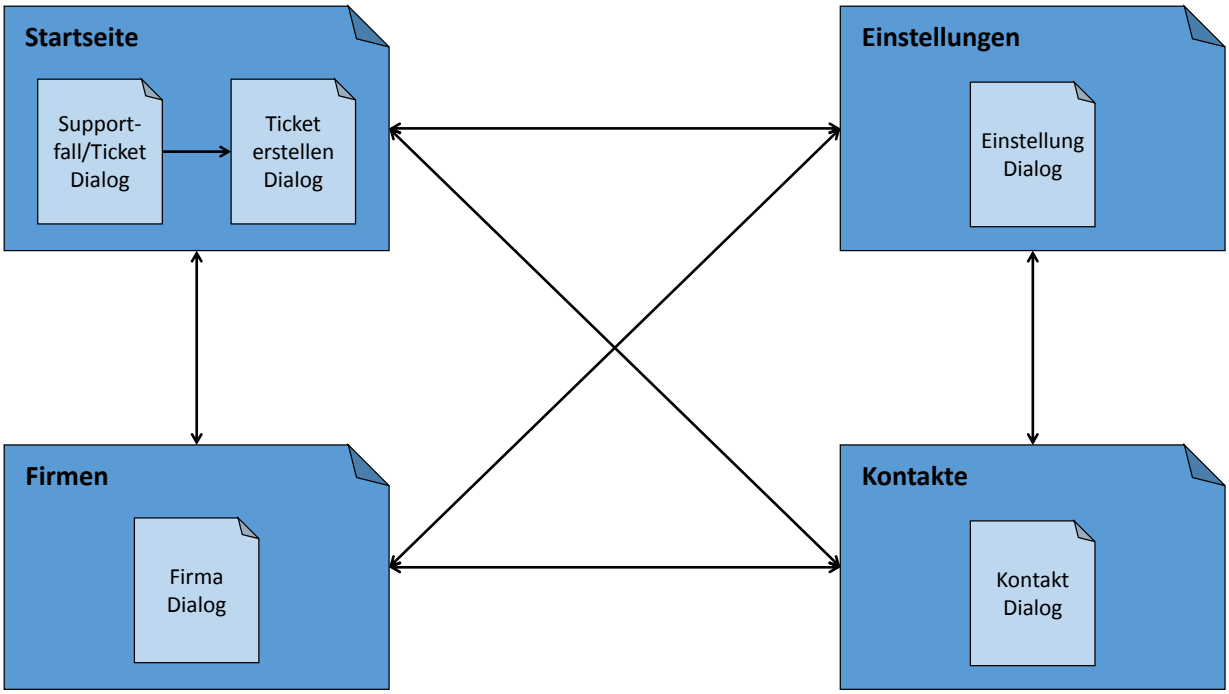

Abbildung 3.3.: Seiten der SharePoint App

Startseite

Die Startseite wird beim Start der App angezeigt. Sie übernimmt die Verwaltung der Supportfälle. Dazu werden Buttons bereitgestellt, mit denen die Supportfälle von Zoho CRM und die Tickets von JIRA abgerufen und in der App gespeichert werden können. Zudem gibt es einen Button mit dem die Supportfälle bei Bedarf nach Zoho CRM zurück synchronisiert werden können. Die in der App gespeicherten Supportfälle werden auf dieser Seite aufgelistet und können nach ihrem Status gefiltert werden. Diese Auflistung enthält nur die wichtigsten Informationen. Durch das Anklicken eines Eintrags wird ein Dialog-Fenster geöffnet, das die gesamten Informationen des Supportfalls zeigt. Bei Bedarf können dort Änderungen am Supportfall vorgenommen werden. Soll dieser dabei der Entwicklung zugewiesen werden, wird ein weiteres Dialog-Fenster geöffnet, das es dem Anwender ermöglicht die Informationen für das Erstellen eines Tickets einzugeben.

Einstellungen

Auf dieser Seite werden die Informationen zur Authentifizierung für JIRA und Zoho CRM verwaltet. Die eingetragenen Informationen werden als Tabelle dargestellt. Per Button können Einträge hinzugefügt, geändert oder gelöscht werden. Für jede dieser Operationen wird ein Dialog-Fenster geöffnet, in dem per Auswahlbox die Anwendung gewählt wird, für die die gewünschte Operation ausgeführt werden soll.

Firmen und Kontakte

Für Firmen und Kontakte gibt es jeweils eine eigene Seite, die die entsprechenden Daten verwaltet. Deshalb gibt es auf beiden Seiten ein Button mit dem die Firmen beziehungsweise Kontakte von Zoho CRM abgerufen und in der App gespeichert werden können. Die gespeicherten Einträge werden auf diesen Seiten als Tabelle aufgelistet. Analog zu den Supportfällen, wird auch hier beim Klick auf einen dieser Einträge ein Dialog-Fenster geöffnet, das die gesamten Informationen zu dem entsprechenden Eintrag anzeigt.

3.4.2. Aufbau der Seiten

Damit sich der Anwender innerhalb der App einfacher zurechtfindet, sollten alle Seiten einheitlich aufgebaut sein. Aus diesem Grund wurden im Rahmen dieser Arbeit Elemente, die auf jeder Seite vorhanden sein sollen und deren Positionierung festgelegt. Das Ergebnis ist in Abbildung 3.4 schematisch dargestellt.

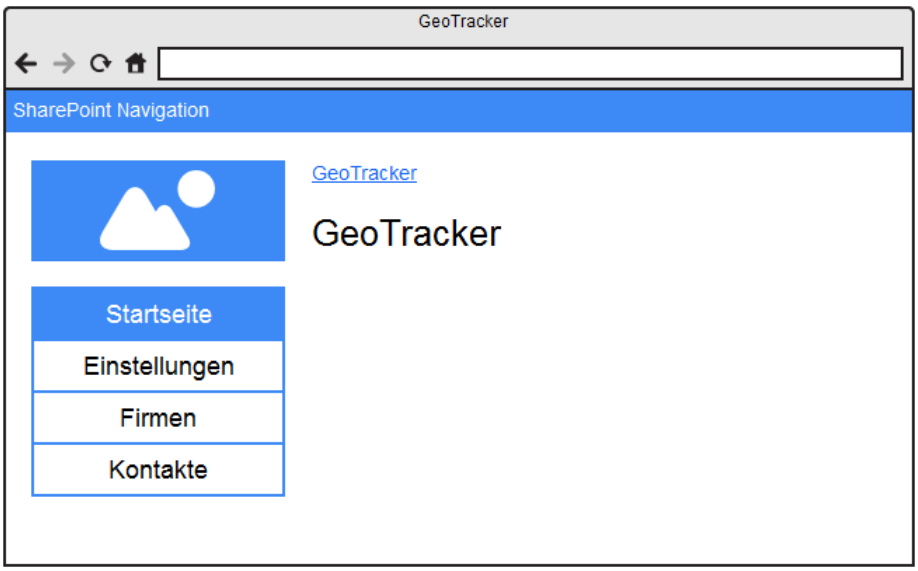

Abbildung 3.4.: Aufbau der Seiten in der SharePoint App

Bei den SharePoint gehosteten Apps werden automatisch bestimmte Designelemente auf jeder Seite platziert. So wird zu Beginn der Seite die SharePoint Navigationsleiste, links darunter das SharePoint Logo und rechts daneben ein Link, der auf die Startseite der App verweist, eingefügt. Der Titel der Seite wird, ebenfalls auf jeder Seite, rechts neben dem Logo angezeigt. Da diese Elemente auf jeder SharePoint Seite vorhanden sind, soll dadurch die Grenze zwischen SharePoint und den Apps verschwimmen. Damit soll eine einheitliche Nutzung gewährleistet werden.

Neben den von SharePoint eingefügten Elementen, werden auch eigene Elemente platziert. Damit der Anwender zwischen den Seiten navigieren kann, soll unter dem Logo eine Navigationsleiste eingeblendet werden. Diese listet alle in der App vorhandenen Seiten auf und zeigt dabei an, auf

welcher Seite sich der Anwender momentan befindet. Der eigentliche Inhalt der Seiten wird rechts daneben platziert. Dabei handelt es sich zumeist um Tabellen, die die von Zoho CRM abgerufen Einträge darstellen.

3.4.3. Workflow

Die App soll, wie im Abschnitt 3.2 beschrieben, die Supportfälle von Zoho CRM abrufen und sie bei sich speichern. Durch die Bearbeitung der Supportfälle in der App soll es möglich sein, dass sie verschiedene Zustände einnehmen können. Beispielsweise soll ein Supportfall der Entwicklung zugewiesen werden können, wodurch in JIRA ein entsprechendes Ticket angelegt wird. Aus diesem Grund wurde im Rahmen dieser Arbeit ein Workflow entwickelt, der festlegt, wann ein Supportfall welchen Zustand einnehmen kann und wann eine Interaktion mit Zoho CRM oder JIRA nötig ist. Die Abbildung 3.5 zeigt diesen Workflow.

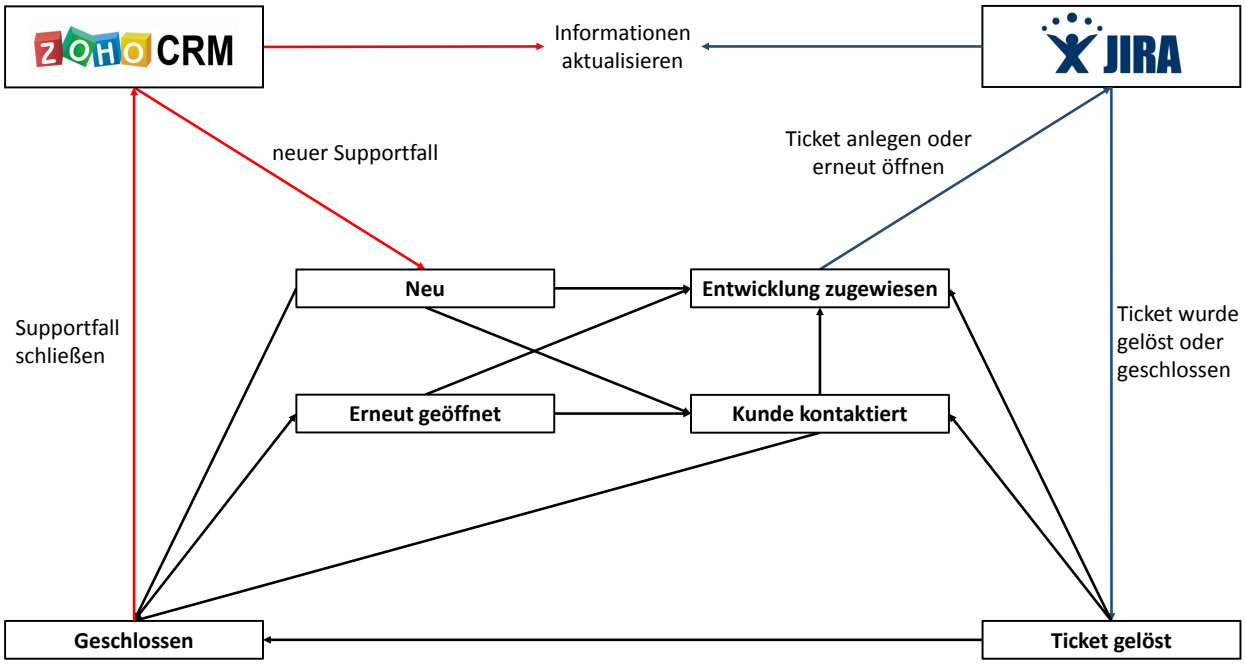

Abbildung 3.5.: Ablauf der Bearbeitung eines Supportfalls

- **Abruf der Supportfälle von Zoho CRM**
  Dazu wird das älteste Änderungsdatum aller Supportfälle der App bestimmt. Anschließend werden damit alle Supportfälle von Zoho CRM abgerufen, die nach diesem Datum erstellt oder geändert wurden. Diese werden nun mit denen aus der App abgeglichen. Existiert der Supportfall in der App bereits, wird dieser, wenn es Änderungen gab, aktualisiert. Anderenfalls wird der Supportfall der App hinzugefügt und bekommt den Status *Neu*.

- **Abruf der Tickets von JIRA**
  Dabei werden nacheinander alle von der App aus angelegten Tickets in JIRA abgerufen. Sind Änderungen aufgetreten, werden die Informationen in der App aktualisiert. Wurde das Ticket gelöst oder geschlossen, wird der Status des Supportfalls, zu dem das Ticket gehört, zu *Ticket gelöst* geändert.

- **Neu**
  Jeder von Zoho CRM abgerufene nicht geschlossene Supportfall wird der App als neuer Supportfall hinzugefügt, falls er dort nicht bereits vorhanden ist.

- **Kunde kontaktiert**
  Dieser Zustand dient dem Anwender als Hinweis, dass mit dem Kunden gesprochen wurde oder dies noch getan werden muss. Falls gewünscht können Kommentare hinzugefügt werden, um die Ergebnisse der Unterhaltung zu dokumentieren.

- **Entwicklung zugewiesen**
  Durch das Zuweisen des Supportfalls zur Entwicklung wird geprüft, ob in JIRA bereits ein entsprechendes Ticket existiert. Ist dies nicht der Fall, wird ein neues Ticket angelegt und die Informationen zum Supportfall gespeichert. Wurde bereits ein Ticket angelegt, wird es noch einmal für die Bearbeitung geöffnet.

- **Ticket gelöst**
  Sobald das Ticket zu einem Supportfall in JIRA gelöst oder geschlossen wurde, wechselt der Supportfall in diesen Zustand.

- **Geschlossen**
  Wird ein Supportfall geschlossen, werden alle Informationen nach Zoho CRM synchronisiert und dort somit ebenfalls geschlossen.

- **Erneut geöffnet**
  Bei Bedarf ist es möglich einen geschlossenen Supportfall nochmals zu öffnen, um eine erneute Bearbeitung zu ermöglichen.

3.5. Kommunikation

Die Hauptaufgabe der App ist der Datenaustausch zwischen Zoho CRM und JIRA und dessen Koordination (siehe Abschnitt 3.4.3). Um dies zu bewerkstelligen, muss die App mit den Anwendungen kommunizieren, um benötigte Daten abzufragen oder zu setzen. Die Kommunikation verläuft über die REST APIs der beiden Anwendungen.

3.5.1. Kommunikation mit Zoho CRM

In diesem Abschnitt wird die Kommunikation der App mit Zoho CRM betrachtet. Dabei werden sowohl die verwendeten Methoden als auch dessen Verwendungszweck innerhalb der App beschrieben.

Daten von Zoho CRM abfragen

Damit die App Supportfälle, Firmen und Kontakte von Zoho CRM darstellen kann, müssen diese von dort abgerufen werden. Dazu stellt die REST API von Zoho CRM die Methode *getRecords* bereit. Mit dieser können die gewünschten Einträge abgefragt werden. Abbildung 3.6 zeigt schematisch den Ablauf.

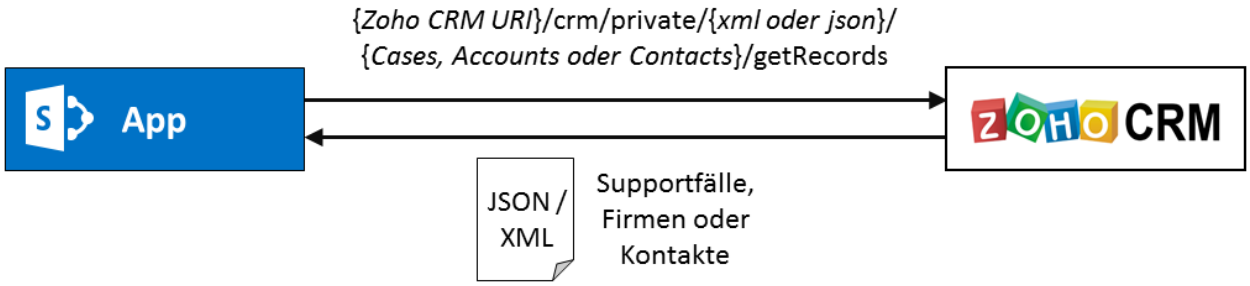

Abbildung 3.6.: Supportfälle, Firmen und Kontakte von Zoho CRM holen

Die URI zu der Abfrage ist ebenfalls in der Abbildung 3.6 zu sehen und setzt sich aus verschiedenen Teilen zusammen. So handelt es sich bei der *Zoho CRM URI* um die URI, die der Anwender in den Einstellungen der App hinterlegt. Durch die Angabe von *xml* oder *json* wird festgelegt, in welchem Format die Einträge zurückgeliefert werden sollen. In der App wird JSON genutzt, aufgrund der einfachen Handhabung innerhalb von JavaScript. Letztlich muss in der URI noch festgelegt werden, welche Daten von Zoho CRM erwartet werden. Diese Anfrage wird von der App genutzt, um Supportfälle, Firmen und Kontakte von Zoho CRM abzurufen.

Zusätzliche Optionen werden per Queryparameter mitgegeben. So muss zu jeder Anfrage der Authentifizierungstoken mitgeschickt werden, damit sie erfolgreich ausgeführt werden kann. Des Weiteren kann ein Datum als zusätzlicher Parameter gesetzt werden, um nur die Einträge zu erhalten, die nach diesem Datum erstellt oder geändert wurden. Dies kann in der App genutzt werden, in dem von allen dortigen Einträgen das älteste Änderungsdatum bestimmt und bei der Anfrage mitgeschickt wird. Dadurch kommen nur die Einträge zurück, die für die App neu sind oder sich in der Zwischenzeit geändert haben.

Die Anzahl der Einträge die Zoho CRM pro Anfrage zurückliefert ist auf 200 beschränkt. Gibt es mehr Einträge, werden weitere Abfragen benötigt. Um nicht nochmals die bereits abgefragten Einträge zu erhalten, gibt es zwei Index Parameter, mit denen festgelegt werden kann, welche Einträge zurückgeliefert werden sollen. So ist es für die App trotz Beschränkung möglich, alle Einträge zu erhalten.

Supportfälle in Zoho CRM aktualisieren

Da Supportfälle von der App aus bearbeitet werden können, muss es möglich sein, diese mit Zoho CRM zu synchronisieren. Zu diesem Zweck kann die *updateRecords* Methode der Zoho CRM API verwendet werden. Mit dieser ist es möglich, ein oder mehrere Einträge zu aktualisieren. Der Ablauf wird in Abbildung 3.7 veranschaulicht.

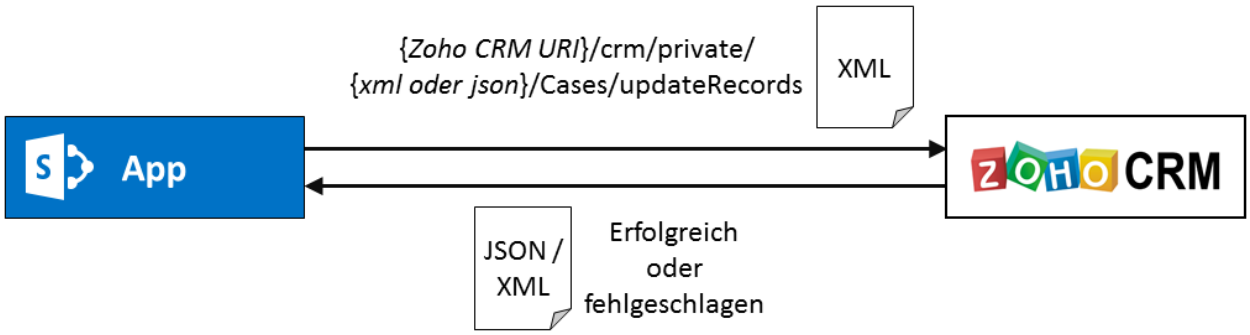

Abbildung 3.7.: Einen oder mehrere Supportfälle aktualisieren

Die für die Abfrage benötigte URI (siehe Abbildung 3.7) ist analog zu der für die *getRecords* Methode (siehe Abbildung 3.6). So beginnt diese auch mit der URI zu Zoho CRM, die in den Einstellungen hinterlegt wird. Das gewünschte Format für die Rückgabe muss ebenfalls angegeben werden. Schließlich wird festgelegt, welche Daten aktualisiert werden sollen. Da die App auf die Firmen und Kontakte nur lesend zugreift, wird die Abfrage nur für das Aktualisieren der Supportfälle genutzt.

Analog zu der *getRecords* Methode, werden bei dieser Methode zusätzliche Optionen ebenfalls per Queryparameter mitgegeben. Neben dem Authentifizierungstoken müssen bei dieser Anfrage die zu aktualisierenden Einträge im XML-Format als Parameter mitgegeben werden. Das XML kann einen oder mehrere Einträge enthalten, wobei es genauso aufgebaut sein muss, wie die XML, die Zoho CRM beispielsweise bei *getRecords* zurückliefern würde.

Grundsätzlich werden bei dieser Anfrage zwei Fälle unterschieden, werden ein oder mehrere Einträge aktualisiert. Bei einem Eintrag muss dessen ID als Parameter mitgegeben werden und die XML darf entsprechend nur einen Eintrag enthalten. Sollen mehrere Einträge mit einer Anfrage aktualisiert werden, muss dies durch setzten eines Parameter angegeben werden. Zudem wird nun zu jedem Eintrag eine ID benötigt, weshalb diese nicht mehr der Anfrage direkt mitgegeben werden, sondern in der XML zu dem jeweiligen Eintrag hinzugefügt werden. Zu beachten ist außerdem, dass maximal 100 Einträge mit einer Anfrage aktualisiert werden können. Die App nutzt beide beschriebenen Varianten. Wird ein Supportfall geschlossen, wird dieser direkt mit Zoho CRM synchronisiert. Zudem soll die App eine Synchronisierung aller Supportfälle ermöglichen. Enthält die App zu diesem Zeitpunkt mehr als 100 Supportfälle, werden entsprechend mehrere Anfragen gesendet.

Die Rückgabe dieser Methode enthält die Information, ob das Aktualisieren erfolgreich war oder nicht. Werden mehrere Einträge mit einer Anfrage bearbeitet, enthält die Rückgabe zu jedem Eintrag diese Information.

3.5.2. Kommunikation mit JIRA

In diesem Abschnitt wird die Kommunikation der App mit JIRA beschrieben. Dazu werden hier analog zur Kommunikation mit Zoho CRM die Methoden genannt und erläutert sowie deren Verwendungszweck innerhalb der App beschrieben.

Ticket Metadaten abfragen

Die App soll es ermöglichen Tickets in JIRA anzulegen, wobei es dem Anwender möglich sein soll bestimmte Felder zu setzen. Da JIRA sehr stark angepasst werden kann, können sich die vorhandenen Felder beziehungsweise dessen mögliche Werte von Projekt zu Projekt unterscheiden. Aus diesem Grund müssen, bevor ein Ticket angelegt werden kann, die Metadaten abgefragt werden. Dazu stellt die JIRA REST API die *createmeta* Methode zur Verfügung. Deren Ablauf wird in Abbildung 3.8 schematisch dargestellt.

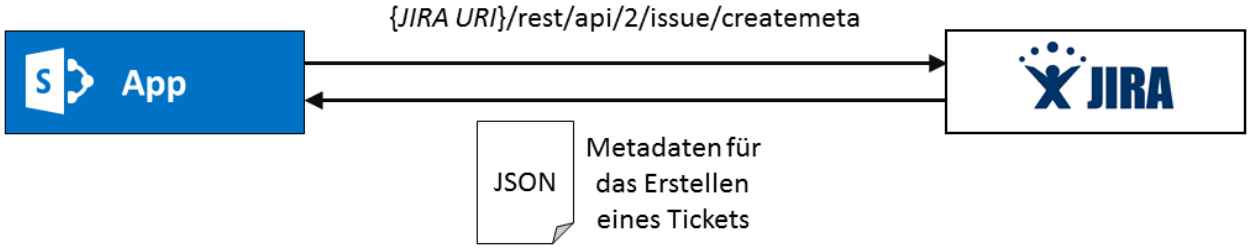

Abbildung 3.8.: Metadaten für das Erstellen eines Tickets abfragen

Für die Anfrage wird die URI zu JIRA benötigt, die in den Einstellungen vom Anwender hinterlegt wird. Wird anschließend die Anfrage mit der in Abbildung 3.8 gezeigten URI ausgeführt, werden zu jedem in JIRA vorhandenen Projekt die möglichen Tickettypen angezeigt. Durch Queryparameter ist es möglich, detailliertere Metadaten abzufragen oder die Rückgabe auf bestimmte Projekte zu beschränken.

Damit die Anfrage erfolgreich ausgeführt werden kann, ist eine Authentifizierung bei JIRA nötig. Dazu stehen grundsätzlich mehrere Methoden zur Verfügung, wobei die meisten nur serverseitig möglich sind. Da in der SharePoint App allerdings nur clientseitiger Code ausgeführt werden kann (siehe Abschnitt 3.3.3), können diese Varianten nicht verwendet werden. Die einzige Möglichkeit sich clientseitig bei JIRA zu Authentifizieren ist, den Benutzernamen und das Passwort per Queryparameter mitzuschicken. Dies hat allerdings den Nachteil, dass diese Informationen als Klartext versendet werden.

Für das Erstellen eines Tickets benötigt die App die detaillierten Metadaten. Die Informationen zu den Tickettypen alleine reichen nicht aus, denn dem Anwender soll es möglich sein, auch weitere Informationen anzugeben. Allerdings kann abhängig von der Anzahl der in JIRA vorhandenen Projekte die Antwort entsprechend groß werden. Dies kann zu Problemen führen, da die App Antworten nur bis zu einer bestimmten Größe akzeptiert. Außerdem würde auch die Performance leiden, da das Parsen der Antwort oder das Auslesen der benötigten Informationen entsprechend aufwendiger werden würde. Aus diesem Grund erfolgt die Abfrage der Metadaten im Rahmen dieser Arbeit in mehreren Schritten, wie das Sequenzdiagramm in Abbildung 3.9 veranschaulicht.

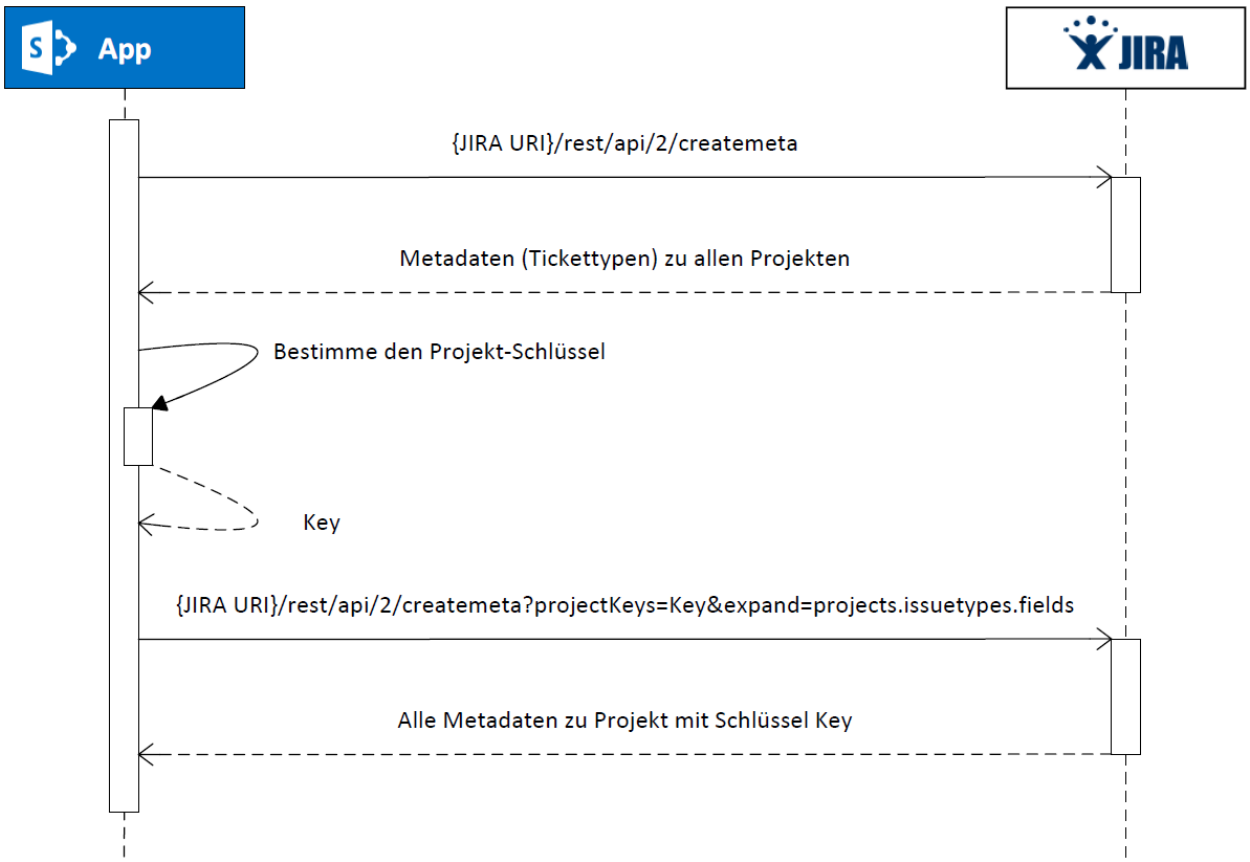

Abbildung 3.9.: Ablauf zur Abfragen aller benötigten Metadaten

Die App kennt den Namen des Projekts, zu dem die Metadaten abgefragt werden sollen, jedoch nicht dessen Schlüssel. Somit können die Metadaten nicht direkt auf ein Projekt beschränken werden, sondern es muss zuvor der Projektschlüssel bestimmt werden. Aus diesem Grund holt sich die App wie in Abbildung 3.9 zu sehen ist zu Beginn die Metadaten von JIRA ohne weitere Angaben. Daraufhin werden von JIRA die Metadaten zu allen Projekten zurückgegeben. Diese Antwort enthält die möglichen Tickettypen zu jedem Projekt sowie deren Schlüssel. Mit diesen Metadaten und mithilfe des Projektnamens wird nun dessen Schlüssel bestimmt. Letztlich macht die App eine weitere Anfrage an JIRA. Dieses Mal wird mithilfe des *projektKeys* Parameters die Antwort auf das entsprechende Projekt beschränkt. Mit dem *expand* Parameter wird zudem angegeben, dass

die Metadaten zu allen Feldern zurückgegeben werden sollen. Dadurch erhält die App schließlich die detaillierten Metadaten zu dem angegebenen Projekt.

Ticket erstellen

Eine der Hauptaufgaben der App ist, dem Anwender die Möglichkeit zu bieten ein Supportfall der Entwicklung zuzuweisen. Dabei soll mit den Informationen des Supportfalls ein Ticket in JIRA erstellt werden. Um dies zu realisieren, werden zuerst die Metadaten abgefragt (siehe Abbildung 3.9), um dem Anwender anzuzeigen, welche Informationen benötigt werden und eventuell eine Auswahl der möglichen Werte zu geben. Das Ticket kann anschließend mit der von der JIRA REST API bereitgestellten *issue* Methode erstellt werden. Die Abbildung 3.10 veranschaulicht dazu den Ablauf.

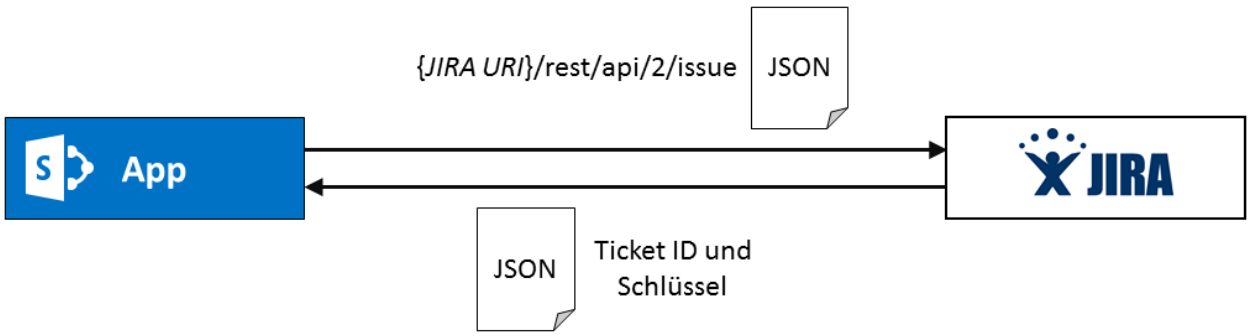

Abbildung 3.10.: JIRA Ticket erstellen

Die JIRA REST API erwartet bei dieser Anfrage zusätzlich ein JSON-Dokument. Dieses muss mindestens alle für das Erstellen eines Tickets erforderlichen Informationen enthalten. Standardmäßig ist das Projekt, der Tickettyp und eine Zusammenfassung anzugeben. Darüber hinaus kann es auch weitere Informationen enthalten. Der Aufbau des JSON-Dokuments entspricht grundsätzlich dem das bei der Abfrage eines Tickets (siehe Abbildung 3.11) zurückliefert wird.

Wurde das Ticket erfolgreich erstellt, liefert JIRA ein JSON-Dokument zurück, in dem die ID und der Schlüssel des neu angelegten Tickets enthalten sind. Dieser Schlüssel wird von der App gespeichert und verwendet, um das Ticket bei Bedarf abzurufen und zu aktualisieren (siehe Abbildung 3.11). Im Fehlerfall wird kein Ticket angelegt und ein JSON-Dokument zurückgeliefert, in dem der Grund dafür genannt wird. Ein solcher Fehler tritt auf, wenn im JSON-Dokument eine erforderliche Information fehlt. Dies wird von der App abgefangen und der Anwender darauf hingewiesen, dass das Ticket nicht erstellt werden konnte.

Ticket abfragen oder aktualisieren

Bei Bedarf soll die App alle von ihr aus erstellten Tickets abrufen und bei Änderungen die Informationen in der App aktualisieren. Außerdem muss die App von Zeit zu Zeit die Tickets in

JIRA aktualisieren, beispielsweise wenn ein Ticket erneut geöffnet werden soll (siehe Abschnitt 3.4.3). Zu diesen beiden Fällen stellt die JIRA REST API eine Methode bereit, deren Ablauf in der Abbildung 3.11 zu sehen ist.

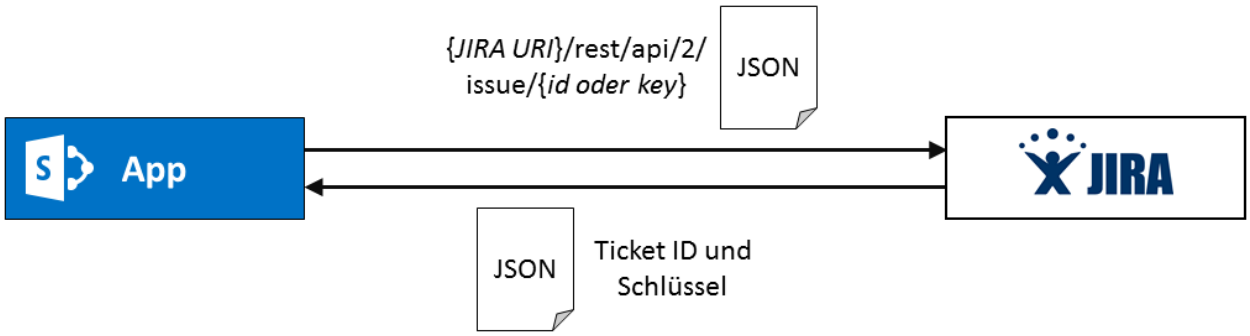

Abbildung 3.11.: Ticket abfragen oder aktualisieren

Für diese Anfrage wird neben der URI zu JIRA auch die ID oder der Schlüssel zu einem Ticket erwartet. Dieser Schlüssel wird von der App zu jedem Ticket gespeichert, nachdem es erstellt wurde. So ist es möglich, jedes von der App aus angelegte Ticket bei Bedarf abzurufen oder zu aktualisieren.

Durch eine HTTP-GET Anfrage an die in Abbildung 3.11 dargestellte URI liefert JIRA alle Informationen zu dem angegebenen Ticket als JSON-Dokument zurück. Mit dessen Hilfe kann die App die Informationen in der App aktualisieren, falls in der Zwischenzeit Änderungen aufgetreten sind.

Durch eine HTTP-POST Anfrage an dieselbe URI ist es möglich, Änderungen am Ticket vorzunehmen. Dazu muss der Anfrage ein JSON-Dokument mit allen gewünschten Änderungen mitgegeben werden. In der App wird diese Methode unter anderem dazu verwendet, ein Ticket erneut zur Bearbeitung zu öffnen oder Kommentare anzufügen. Als Antwort liefert JIRA lediglich ein Statuscode zurück, der angibt, ob die Änderungen erfolgreich waren oder nicht. Dieser Statuscode wird von der App ausgewertet und im Fehlerfall ein entsprechender Hinweis ausgegeben.

4. Implementierung

Dieser Abschnitt befasst sich mit der Implementierung der App, die im Rahmen dieser Arbeit entwickelt wurde. Dabei wird zuerst die Speicherung der Daten innerhalb der App beschrieben. Anschließend wird darauf eingegangen, wie die Listen und deren Inhalt dargestellt werden. Dazu werden unter anderem die Dialog-Fenster betrachtet, die von SharePoint bereitgestellt werden. Darüber hinaus wird beschrieben, was bei Anfragen an externe Webservices zu beachten war und wie die App lokalisiert wurde.

4.1. Speicherung der Daten

In diesem Abschnitt wird auf die Speicherung der Daten innerhalb der App eingegangen. Dazu werden zu Beginn die im Rahmen dieser Arbeit verwendeten SharePoint Listen genauer betrachtet. Anschließend wird in den darauf folgenden Abschnitten deren Verwendung innerhalb der App beschrieben.

4.1.1. SharePoint Listen

Die einzige Möglichkeit in SharePoint Apps Daten persistent zu speichern sind SharePoint Listen. Diese werden in SharePoint allgemein zur Speicherung verschiedenster Informationen und Dateien eingesetzt. Dabei entsprechen der Aufbau und die Funktionsweise dieser Listen verallgemeinert dem von Datenbanktabellen. So können, analog zu Spalten, Felder definiert werden, zu denen unter anderem der Typ der zu speichernden Daten festgelegt werden kann. Diese Definition erfolgt mittels einer XML-Datei, in der alle Felder festgelegt werden, die eine SharePoint Liste besitzen soll. Das Listing 4.1 zeigt beispielhaft, wie eine solche Felddefinition aussieht.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <Elements xmlns="http://schemas.microsoft.com/sharepoint/">
3    <Field
4      ID="{67faa778-63e7-49ad-ae5c-5e942786dd57}"
5      Name="SharePoint_Status"
6      Type="Text"
7      Required="TRUE">
8    </Field>
9    <!-- weitere Felder -->
10 </Elements>
```

Listing 4.1: Definition eines Felds einer SharePoint Liste

Um ein Feld zu definieren, müssen, wie in Listing 4.1 zu sehen ist, eine *ID* und ein *Name* festgelegt werden. Diese müssen in der gesamten App eindeutig sein, da mithilfe dieser Attribute der Zugriff auf das entsprechende Feld erfolgt. Mittels des *Type* Attribut wird festgelegt, welche Daten in dem Feld gespeichert werden können. Im Beispiel können in dem Feld beliebige Unicode-Zeichen enthalten sein, allerdings ist deren Anzahl begrenzt. Falls dies nicht ausreichend sein sollte, ist es mithilfe des Typs *Note* möglich, eine größere Anzahl zu speichern. Mittels des *Required* Attributs wird festgelegt, ob das Feld beim Anlegen eines Listeneintrags spezifiziert werden muss oder ob die Angabe optional ist. Mit diesen Mitteln können beliebige SharePoint Listen definiert und angelegt werden, die sich an die eigenen Anforderungen anpassen lassen.

Nachdem eine SharePoint Liste wie beschrieben angelegt wurde, können Einträge sowohl manuell als auch programmatisch hinzugefügt, bearbeitet oder gelöscht werden. Um die Einträge einer Liste auszulesen und deren Informationen auszuwerten, sind einige Schritte notwendig, die in Listing 4.2 aufgezeigt werden.

```
1  var ctx = SP.ClientContext.get_current();
2  var web = ctx.get_web();
3  var someList = web.get_lists().getByTitle('Titel der Liste');
4  var listItems = someList.getItems(new SP.CamlQuery());
5  ctx.load(listItems);
6  ctx.executeQueryAsync(function () {
7      // Erfolgreich!
8  },
9  function () {
10     // Fehlgeschlagen!
11 });
```

Listing 4.2: Laden einer SharePoint Liste und dessen Elementen

Die gewünschte Liste kann über ihren Namen, wie in Listing 4.2 zu sehen ist, oder über ihre ID abgefragt werden. Um nun die einzelnen Elemente der Liste zu laden, wird die *getItems* Methode der Liste aufgerufen. Dieser wird eine CAML (Collaborative Application Markup Language) Abfrage übergeben. CAML Abfragen haben eine XML-ähnliche Syntax und erfüllen einen ähnlichen Zweck wie SQL bei Datenbanken. Es können damit die Elemente, die zurückgeliefert werden sollen, nach bestimmten Kriterien selektiert werden. Im Beispiel wird eine leere CAML Anfrage übergeben. Dadurch werden alle Elemente die momentan in der Liste enthalten sind als Collection zurückgeliefert. Bevor mit den Elementen gearbeitet werden kann, müssen sie in den aktuellen Kontext geladen werden. Ist dies erfolgreich, kann über die Collection iteriert und so die Informationen der einzelnen Elemente ausgewertet oder bearbeitet werden.

4.1.2. Speicherung der Einstellungen

Für die Einstellungen wurde im Rahmen dieser Arbeit eine eigene SharePoint Liste definiert und angelegt. Darin werden alle Informationen gespeichert, die zur Authentifizierung bei Zoho CRM beziehungsweise JIRA benötigt werden. Die Liste umfasst deshalb die Felder *Name* und *URL*, die für beide Anwendungen notwendig sind, sowie den *Authentifizierungstoken* für Zoho CRM bezie-

hungsweise *Benutzername* und *Passwort* für JIRA. Diese Informationen werden vom Anwender eingegeben und jedes Mal ausgelesen, wenn die App sie für Anfragen benötigt. Da es für Passwortfelder keinen eigenen Typ gibt, der bei der Felddefinition angegeben werden kann, müssen der Authentifizierungstoken und das Passwort im Klartext in der Liste gespeichert werden.

4.1.3. Speicherung der Firmen- und Kontaktdaten

Im Rahmen dieser Arbeit wurde für die Firmen und Kontakte jeweils eine eigene Liste definiert und angelegt. Die Felder wurden an die entsprechenden Felder von Zoho CRM angelehnt, sodass die benötigten Informationen in der App gespeichert werden können. Für jede Firma beziehungsweise jeden Kontakt in Zoho CRM, der in der App noch nicht vorhanden ist, wird der entsprechenden Liste ein neues Element hinzugefügt. Falls ein Eintrag bereits vorhanden ist, wird das betreffende Element bei Änderungen aktualisiert. Aufgrund der Anforderungen (siehe Abschnitt 3.2) muss die App nicht das Erstellen, Bearbeiten oder Löschen eines solchen Eintrags ermöglichen, da ein rein lesender Zugriff auf die Firmen und Kontakte von Zoho CRM erfolgt.

4.1.4. Speicherung der Supportfall- und Ticket-Informationen

In der App müssen sowohl die Supportfälle von Zoho CRM, als auch die Tickets von JIRA, die zu den Supportfällen angelegt wurden, gespeichert werden können. Zudem ist es notwendig, die Zuordnung von Supportfall zu Ticket zu speichern. Grundsätzlich gibt es zwei mögliche Lösungsansätze.

- **Separate Listen**
  Hierbei werden für die Supportfälle und Tickets jeweils eigene Listen angelegt. Auch die Zuordnung wird in einer separaten Liste gespeichert. Diese enthält den Status (siehe Abschnitt 3.4.3) und Verweise auf einen Supportfall und falls vorhanden auf ein Ticket. Durch indizierte Zugriffe und JOIN-Operationen können die gesamten Informationen ausgelesen werden.

- **Eine Liste**
  In diesem Fall werden die Supportfälle und Tickets in einer Liste gespeichert. Da zu jedem Supportfall das entsprechende Ticket gespeichert wird, ist eine eigene Liste für die Zuordnung nicht notwendig. Wird zu einem Supportfall kein Ticket angelegt, bleiben die entsprechenden Felder leer. Somit enthält ein Eintrag dieser Liste sämtliche Informationen.

Im Rahmen dieser Arbeit wurde die zweite Variante gewählt und die Supportfall- und Ticket-Informationen in eine Liste gespeichert. Diese Möglichkeit wurde gewählt, da bei der anderen Variante drei Listen und im schlimmsten Fall dreimal so viele Einträge in den Kontext geladen werden müssen. Außerdem sind so keine indizierten Zugriffe oder JOIN-Operationen notwendig.

Um alle Informationen auszulesen, muss somit nur auf eine Liste zugegriffen werden. Auch das Einfügen oder Ändern eines Eintrages wird dadurch erleichtert, da die Änderungen nur eine Liste betreffen.

Anpassung der Zoho CRM Felder

Durch die Speicherung der Supportfall- und Ticket-Informationen in einer Liste entstehen viele leere Einträge, wenn zu einem Supportfall kein Ticket angelegt wird. Um dies weitestgehend zu vermeiden, wurden im Rahmen dieser Arbeit die Supportfall-Felder von Zoho CRM an die Ticket-Felder von JIRA angepasst, sodass für Felder mit gleichem oder ähnlichem Inhalt nur eine Spalte in der Liste benötigt wird, um die entsprechende Information zu speichern. Dazu wurden einige Zoho CRM Felder auf die von JIRA abgestimmt, wodurch eine Art Mapping zwischen diesen entsteht. So enthält der *Betreff* bei Zoho CRM in etwa dieselbe Information, wie die *Zusammenfassung* bei JIRA oder die *Beschreibung* für den Supportfall und das Ticket sind identisch. Neben anderen Feldern wurden auch die *Priorität* und der *Typ* beider Anwendungen aneinander angeglichen. Dazu wurden die entsprechenden Felder in Zoho CRM so angepasst, dass dieselbe Auswahl wie in JIRA möglich ist. Außerdem wurde in Zoho CRM der *Status* angepasst, um eine erneute Bearbeitung eines Supportfalls zu ermöglichen.

Durch das Schließen eines Supportfalls soll die App alle Informationen nach Zoho CRM zurück synchronisieren, wodurch sie dort archiviert werden. Dadurch soll auch im Nachhinein noch nachvollzogen werden können, welche Supportfälle aufgetreten sind und wie sie behoben wurden. Um eine vollständige Nachvollziehbarkeit zu erreichen, sollen bei der Synchronisierung eines Supportfalls auch Ticket-Informationen übertragen werden. Aus diesem Grund wurden in Zoho CRM Felder für die wichtigsten JIRA Informationen angelegt, die für eine spätere Rückverfolgung wichtig sind. So wurden beispielsweise Felder für Versionsnummern angelegt, um nachverfolgen zu können, in welcher Version ein bestimmtes Problem aufgetreten ist und in welcher es behoben wurde. Diese Felder können auch für die Übertragung der Information hin zur App genutzt werden. So kann beispielsweise beim Erstellen eines Supportfalls bereits die Version angegeben werden, auf die sich der Supportfall bezieht. Wird dieser Supportfall der Entwicklung zugewiesen, wird dadurch automatisch auch die angegebene Version an JIRA übertragen.

Eine Auflistung aller Änderungen an den Feldern der Supportfälle bei Zoho CRM ist im Anhang in Tabelle A.1 zu finden. Dort werden außerdem alle Spalten aufgelistet, die die SharePoint Liste für die Speicherung der Supportfall- und Ticket-Informationen enthält.

Optimierung der Performance

Die Anzahl der in der App gespeicherten Supportfälle wird im laufenden Betrieb fortwährend zunehmen. Dabei werden vor allem die geschlossenen Supportfälle einen erheblichen Anteil ausmachen, da ein Supportfall beim Schließen nicht gelöscht wird. Diese Supportfälle werden dabei

nur in Ausnahmefällen erneut zur Bearbeitung geöffnet. Die Liste muss allerdings oftmals durch iteriert werden, um beispielsweise die offenen Supportfälle zu aktualisieren oder um ihren Inhalt darzustellen (siehe Abschnitt 4.2). Dies geht bei zunehmender Anzahl an Listenelementen verstärkt zulasten der Performance, die gerade bei Webanwendungen eine wichtige Rolle spielt.

Da sich die Anzahl der in Bearbeitung befindlichen Supportfälle in einem überschaubaren Rahmen hält und gerade diese für den Anwender besonders interessant sind und deswegen oft bearbeitet werden, wurden für diese eine eigene Liste angelegt. Somit gibt es separate Listen für die offenen und für die geschlossenen Supportfälle. Das Aktualisieren und Darstellen der offenen Supportfälle ist somit deutlich performanter, da sich in der entsprechenden Liste weniger Einträge befinden. Darüber hinaus werden die geschlossenen Supportfälle nun nur noch in den Kontext geladen und durch iteriert, wenn der Anwender diese explizit anzeigt bekommen möchte. Für das Schließen und das erneute Öffnen eines Supportfalls werden nun zwei Operationen benötigt. So muss ein Eintrag aus der einen Liste gelöscht und der anderen hinzugefügt werden. Dies geschieht im Hintergrund, sodass es den Schein hat, es würde sich um eine Liste handeln.

4.2. Grafische Darstellung der Listeninhalte

Dieser Abschnitt befasst sich mit der Darstellung der SharePoint Listen und deren Inhalten. Dazu wird zu Beginn beschrieben, wie diese Listen standardmäßig von SharePoint dargestellt werden und aus welchen Gründen diese Ansicht für die zu entwickelnde App nicht verwendet werden konnte. Anschließend wird beschrieben, welche Alternative für die Darstellung der Listen in der App verwendet wurde und wie diese bei den verschiedenen Listen zum Einsatz kam.

4.2.1. Standarddarstellung von SharePoint

SharePoint legt zu jeder erstellten Liste standardmäßig eigene Seiten für die wichtigsten Operationen an. Dabei handelt es sich unter anderem um eine Seite für die Darstellung der Listeninhalte und Seiten zum Erstellen beziehungsweise Bearbeiten von einzelnen Elementen. Diese Seiten sind für Listen innerhalb einer App unter einer speziellen URL verfügbar: https://*{SharePoint URL}*/*{Name der App}*/Lists/*{Name der Liste}*. Durch die Bereitstellung dieser Seiten soll die Verwendung von SharePoint Listen erleichtert werden, da diese Standardoperationen nicht selbst implementiert werden müssen.

Für die im Rahmen dieser Arbeit entwickelte App konnten diese Seiten allerdings nicht verwendet werden, da diese dem Anwender zu viele Rechte gewähren und die App somit keine Kontrolle mehr über den Inhalt der Listen hätte. In der App soll der Anwender selbst, abgesehen von den Einstellungen, keine Möglichkeit haben Listenelemente hinzufügen oder zu löschen. Diese Operationen sollen ausschließlich von der App ausgeführt werden können. Außerdem soll der Anwender nur bestimmte Informationen eines Eintrags bearbeiten können, wobei er darauf hingewiesen wer-

den soll, falls er eine ungültige Eingabe getätigt hat. Dazu muss die App die Möglichkeit haben die Eingaben zu prüfen, bevor sie in die Liste gespeichert werden, was jedoch mithilfe der von SharePoint bereitgestellten Seiten nicht möglich ist. Darüber hinaus ist die Darstellung der Liste durch SharePoint unübersichtlich, da alle Spalten der Liste auf einmal dargestellt werden. Der Anwender würde dadurch leicht den Überblick verlieren. Aus diesem Grund sollen in der App nur die wichtigsten Spalten zu sehen sein. Erst wenn dies explizit gefordert wird, sollen alle Informationen zu einem Eintrag angezeigt werden.

Aus diesen Gründen musste im Rahmen dieser Arbeit für die Darstellung der Listen und deren Inhalte eine geeignete Alternative gewählt werden. Außerdem mussten die entsprechenden Operationen wie Hinzufügen, Bearbeiten oder Löschen entsprechend den Anforderungen neu implementiert werden (siehe Abschnitt 4.3.1).

4.2.2. Tabellarische Darstellung

Im Rahmen dieser Arbeit wurden die Inhalte der SharePoint Listen als Tabellen grafisch aufbereitet. Dies ist zum einen darauf zurückzuführen, dass die Listen ähnlich wie Tabellen aufgebaut sind und die Darstellung somit ohne größere Umstände möglich ist. Darüber hinaus bieten Tabellen den Vorteil große Mengen an Informationen übersichtlich darzustellen und sind somit für den Anwender auch ohne Erläuterung einfach zu verstehen.

Darstellung der Einstellungen

Die Einstellungen der App enthalten maximal zwei Einträge, einen für Zoho CRM und einen für JIRA. Aus diesem Grund ist eine Tabelle, bei der die Informationen zeilenweise hinzugefügt werden, für die Einstellungen nicht geeignet. Im Rahmen dieser Arbeit wurde aufgrund dessen eine spaltenweise Anordnung der Informationen der Anwendungen gewählt. Dadurch ist die Tabelle übersichtlicher, da alle Informationen zu einer Anwendung untereinander stehen.

Die Tabelle für die Einstellungen wird dynamisch erstellt, da sich die Anzahl der Einträge und deren Inhalt fortwährend ändern kann. Tabellen können mittels jQuery (siehe Abschnitt 2.5.1) um Zeilen, jedoch nicht um Spalten erweitert werden, wie es für die Einstellungen nötig wäre. Aus diesem Grund wird die Tabelle vollständig dynamisch erzeugt. Das bedeutet auch die „Zeilenüberschriften“ werden bei jeder Generierung der Tabelle dynamisch hinzugefügt.

Die Informationen sind in der SharePoint Liste als normaler Text gespeichert. Dies ist bei der Darstellung nicht bei jeder Information wünschenswert. Um die Benutzerfreundlichkeit zu erhöhen, werden bei der Generierung der Tabelle aus diesem Grund die URLs der Anwendungen als Links und das Passwort beziehungsweise der Authentifizierungstoken nicht als Klartext, sondern durch mehrere „*“ dargestellt. Außerdem werden Informationen, die nur eine der beiden Anwendungen betreffen nur bei dieser dargestellt. Bei der anderen wird durch „-“ kenntlich gemacht, dass die Information für diese Anwendung nicht relevant ist.

Darstellung der Supportfälle, Firmen und Kontakte

Die SharePoint Listen für Supportfälle, Firmen und Kontakte wurden im Rahmen dieser Arbeit ebenfalls tabellarisch dargestellt. Im Gegensatz zu den Einstellungen erfolgt die Darstellung hier zeilenweise, denn in diesen Listen können sich beliebig viele Einträge befinden. Dadurch kann ein Eintrag einfach dynamisch als neue Zeile zu der Tabelle hinzugefügt werden. Außerdem kann der Tabellenkopf und somit die Spalten statisch festgelegt werden und müssen nicht bei jedem Erstellen der Tabelle wiederholt hinzugefügt werden. Dadurch ergibt sich für den Aufbau der Tabellen ein Grundgerüst, das in Listing 4.3 am Beispiel der Tabelle für Supportfälle zu sehen ist.

```
<table id="caseIssueHead">
    <thead>
        <tr>
            <th id="rowNoName" class="rowNo"></th>
            <th id="statusName" class="status"></th>
            <th id="subjectName" class="subject"></th>
            <th id="projectName" class="project"></th>
            <th id="typeName" class="type"></th>
            <th id="prioName" class="priority"></th>
            <th id="dueDateName" class="dueDate"></th>
        </tr>
    </thead>
</table>
<div id="containerBody">
    <table id="caseIssueBody"></table>
</div>
```

Listing 4.3: Grundgerüst der Tabelle für Supportfälle

Wie im Listing 4.3 zu sehen ist, werden nicht alle Informationen der Listen in den Tabellen dargestellt. So soll erreicht werden, dass die Tabellen übersichtlich bleiben und die wichtigsten Informationen auf einen Blick ersichtlich sind. Werden zu einem Eintrag die gesamten Informationen benötigt, kann durch einen Klick auf den entsprechenden Eintrag in der Tabelle ein Dialog-Fenster (siehe Abschnitt 4.3) geöffnet werden, indem detailliert alle Informationen dargestellt werden. Diese Klick-Events werden ebenfalls dynamisch zu jedem Eintrag hinzugefügt.

Eine weitere Maßnahme die Tabellen einfach und übersichtlich zu gestalten ist, dass maximal 20 Einträge auf einmal angezeigt werden, wobei sich die Zeilen abwechselnd farblich voneinander unterscheiden. Bei weniger Einträgen werden aus ästhetischen Gründen die Tabellen mit leeren Zeilen aufgefüllt. Wird die Anzahl überschritten, sollen die Zeilen der Tabellen durchgeblättert werden können. Dabei soll nur der Inhalt der Tabelle scrollbar sein, der Tabellenkopf soll seine Position nicht ändern. Dies wurde im Rahmen dieser Arbeit realisiert, indem intern zwei Tabellen verwendet wurden. Wie das Listing 4.3 veranschaulicht, enthält die erste Tabelle lediglich die Spaltennamen. Die Zweite ist von einem *div* Element umschlossen und enthält zu Beginn keine Elemente. Um nun nur den Tabelleninhalt scrollbar zu machen, wurde in der zugehörigen CSS-Datei die Größe des *div* Elements festgelegt und das Attribut *overflow-y* auf *auto* gesetzt. Bei einem übergroßen Inhalt wird dadurch nur ein Ausschnitt davon angezeigt. Der weitere Inhalt ist nur durch Scrollen erreichbar. Um zu veranschaulichen, dass nicht der gesamte Inhalt dargestellt wer-

den konnte, wird automatisch eine entsprechende Bildlaufleiste eingeblendet. Da das *div* Element eine Tabelle enthält, kann durch die Einstellungen am umschließenden Element der Tabelleninhalt durchgeblättert werden.

Um den Schein zu erwecken, dass es sich nur um eine einzige Tabelle handelt, müssen die Spalten beider Tabellen die gleiche Breite haben. Dies wird dadurch erreicht, dass in jeder Zeile die entsprechenden Spalteninhalte das gleiche *class* Attribut spezifizieren. In der CSS-Datei kann darüber eine einheitliche Größe der jeweiligen Spalten festgelegt werden.

Da die Anzahl der Einträge sehr groß werden kann, ist es mithilfe von Auswahlboxen möglich die Einträge, die in den Tabellen angezeigt werden, nach bestimmten Kriterien zu selektieren. Bei den Firmen können die Einträge so nach dem Firmentyp und die Supportfälle nach ihrem Status gefiltert werden. Für die Supportfälle werden für die Speicherung wie im Abschnitt 4.1.4 beschrieben intern zwei Listen verwendet. Um den Schein zu erwecken, es handelt sich nur um eine Liste, wird standardmäßig die Liste für die offenen Supportfälle in der Tabelle dargestellt. Erst nachdem der Anwender über die Auswahlbox auswählt, dass er die geschlossenen Supportfälle angezeigt bekommen möchte, wird die andere Liste geladen und deren Einträge in derselben Tabelle angezeigt.

4.3. Dialog-Fenster

Die Darstellung aller Informationen eines Supportfalls, einer Firma oder eines Kontakts, das Erstellen eines JIRA Tickets, sowie das Hinzufügen, Bearbeiten und Löschen von Listeneinträgen der Einstellungen wurde im Rahmen dieser Arbeit mithilfe der Dialog-Fenster von SharePoint umgesetzt. Dabei handelt es sich um Fenster innerhalb der aktuellen Seite, die geöffnet oder geschlossen werden können, ohne dass der Anwender auf eine neue Seite weitergeleitet wird. Innerhalb des Fensters wird dabei eine eigenständige Seite geladen und ausgeführt. Dadurch können separate Aufgaben bearbeitet werden, die keinen direkten Einfluss auf die unterliegende Seite haben. Neben diesen Eigenschaften war auch die Möglichkeit durch eigenständige Fenster die Inhalte übersichtlich zu gestalten, ausschlaggebend für die Entscheidung Dialog-Fenster innerhalb der App einzusetzen.

SharePoint bietet die Möglichkeit die Dialog-Fenster über Optionen gezielt an die eigenen Bedürfnisse anzupassen. Die meisten der dabei möglichen Angaben sind optional, wie beispielsweise die Größe. Andere Angaben dagegen sind notwendig, damit der Dialog erstellt werden kann. So muss die URL zu der Seite spezifiziert werden, die in dem Dialog-Fenster angezeigt werden soll. Zudem muss auch der Titel des Fensters festgelegt werden. Die so gesetzten Optionen werden letztlich der Methode übergeben, die mit deren Hilfe einen entsprechenden Dialog erstellt. Das Listing 4.4 veranschaulicht an einem Beispiel den beschriebenen Mechanismus.

```
var option = {
    width: 400,
    height: 200,
    url: '../Pages/Seite.aspx',
    title: 'Titel',
    args: JSON.stringify(args),
    dialogReturnValueCallback: function (result, returnValue) {
        // Wird ausgeführt, nachdem der Dialog geschlossen wurde!
    }
};

SP.UI.ModalDialog.showModalDialog(option);
```

Listing 4.4: Öffnen eines Dialog-Fensters

Der Datenaustausch zwischen der ursprünglichen Seite und dem Dialog-Fenster wird ebenfalls über die Optionen festgelegt. So können über das *args* Attribut dem Dialog zusätzlich Parameter übergeben werden. Diese werden beim Aufruf der Seite mitgeschickt und müssen im JSON-Format vorliegen, damit sie übertragen werden können. Im Dialog können diese Parameter abgefragt und anschließend nach Belieben verwendet werden. Neben Parametern, die dem Dialog übergeben werden, kann auch der Dialog etwas an die Seite zurückgeben, von der der Aufruf erfolgte. Dazu wird in den Optionen eine Callback-Methode spezifiziert, die ausgeführt wird, sobald der Anwender das Dialog-Fenster schließt.

4.3.1. Listenoperationen für die Einstellungen

SharePoint legt für jede Liste standardmäßig eigene Seiten für das Hinzufügen, Bearbeiten und Löschen von Listenelementen an. Wie im Abschnitt 4.2 bereits beschrieben wurde, können diese Seiten für die zu entwickelnde App allerdings nicht verwendet werden, da die App dadurch keine Kontrolle über den Inhalt der Liste hat. Aus diesem Grund mussten die Listenoperationen für die Einstellungen im Rahmen dieser Arbeit neu entwickelt werden. Für diesen Zweck wurden Dialog-Fenster eingesetzt, damit der Anwender nicht bei jeder Listenoperation auf eine neue Seite weitergeleitet wird.

Für alle benötigten Listenoperationen wurde nur ein Dialog-Fenster verwendet. Die Unterscheidung, welche Operation ausgeführt wird, erfolgt über einen Parameter, der dem Dialog beim Erstellen übergeben wird. Zudem erhält der Dialog entsprechend der auszuführenden Aktion einen passenden Titel. Dadurch muss intern nur eine Seite erstellt werden, die anhand des übergebenen Parameters die gewünschte Operation ausführt.

Unabhängig davon, welche Operation ausgeführt werden soll, werden zu Beginn die Listenelemente in den Kontext geladen. Beim Hinzufügen wird anhand derer bestimmt, zu welcher Anwendung noch kein Eintrag hinzugefügt wurde. Dadurch soll sichergestellt werden, dass den Einstellungen zu einer Anwendung nur ein Eintrag hinzugefügt werden kann. Gibt es zu jeder der Anwendung bereits einen Eintrag ist es entsprechend nicht möglich einen weiteren hinzuzu-

fügen. Beim Bearbeiten werden die Listenelemente verwendet, um die Eingabefelder zu befüllen, mit denen der Anwender Änderungen an dem Eintrag vornehmen kann. Analog dazu werden die Felder beim Löschen eines Eintrags ebenfalls befüllt, jedoch ist es dabei nicht möglich, diese zu ändern. Die Informationen werden beim Löschen angezeigt, damit der Anwender sich vergewissern kann, dass dieser Eintrag wirklich gelöscht werden soll. Die Dialog-Fenster zum Bearbeiten oder Löschen können nur geöffnet werden, wenn es mindestens einen Eintrag in der Liste gibt.

Wurde erfolgreich ein Eintrag hinzugefügt, bearbeitet oder gelöscht, wird auf der Seite für die Einstellungen, die Tabelle zur Darstellung der Listeninhalte neu erstellt und somit die angezeigten Informationen aktualisiert.

4.3.2. Detaillierte Darstellung von Supportfällen, Firmen und Kontakten

Für das Darstellen aller Informationen eines Supportfalls, einer Firma oder eines Kontakts wurden im Rahmen dieser Arbeit ebenfalls Dialog-Fenster verwendet. Diese werden geöffnet, nachdem ein Eintrag aus der entsprechenden Tabelle angeklickt wurde.

Damit ein Eintrag angezeigt werden kann, muss er aus der Liste ausgelesen werden. Dazu muss der Eintrag eindeutig identifiziert werden können. In Zoho CRM werden beim Erstellen eines Supportfalls, einer Firma oder eines Kontakts bereits automatisch zur Identifizierung eindeutige IDs vergeben. Diese werden beim Abrufen der Daten von Zoho CRM in der App zu jedem Eintrag gespeichert und dort ebenfalls für die Identifizierung verwendet. Dadurch muss in der App keine eigene ID erstellt und verwaltet werden. Diese ID wird dem Dialog übergeben, wodurch dort der entsprechende Eintrag in der Liste gefunden und angezeigt werden kann.

Nachdem ein Eintrag aus der Liste ausgelesen wurde, sollen im Dialog-Fenster alle Informationen zu diesem dargestellt werden. Da der Anwender leicht die Übersicht verlieren würde, wenn die Informationen unstrukturiert angezeigt werden, wurden Kategorien gebildet. Diese können nach Bedarf ein- und ausgeblendet werden. Für Supportfälle wurden die Informationen in die Kategorien „Allgemeine Informationen“, „Supportfall-Informationen“ und „Ticket-Informationen“ eingeteilt. Dabei werden Letztere nur angezeigt, wenn entsprechend zu einem Supportfall ein JIRA Ticket erstellt wurde. Bei den Firmen und Kontakten gibt es zwei Kategorien, „Allgemeine Informationen“ und „Adressen“. Durch diese Einteilung ist eine übersichtliche und strukturierte Darstellung der Informationen möglich.

Im Gegensatz zu den Firmen und Kontakten, bei denen die Informationen nur angezeigt werden, ist bei den Supportfällen auch eine Bearbeitung möglich. Dabei können nicht alle Informationen angepasst werden, sondern nur bestimmte. Darunter fallen unter anderem das Fälligkeitsdatum, die E-Mail-Adresse oder der Status des Supportfalls. Die Eingaben werden bevor die Änderungen gespeichert werden auf ihre Korrektheit geprüft. Bei der Prüfung wird beispielsweise nachgeschaut, ob ein gültiges Datum oder eine gültige E-Mail-Adresse eingegeben wurde oder ob Felder die nicht leer sein dürfen aufgefüllt wurden. Nur wenn alle Änderungen in Ordnung waren, werden diese

gespeichert. In diesem Fall wird nach dem Schließen des Dialog-Fensters auch die Tabelle zur Darstellung der Supportfälle aktualisiert. Wird beim Bearbeiten eines Supportfalls der Status auf „Entwicklung zugewiesen“ geändert, wird ein neues Dialog-Fenster geöffnet, mit dem ein JIRA Ticket erstellt werden kann.

4.3.3. Erstellen eines JIRA Tickets

Das Erstellen eines JIRA Tickets erfolgt ebenfalls mithilfe eines Dialog-Fensters. Dieses wird geöffnet, sobald ein Supportfall der Entwicklung zugewiesen wird, allerdings nur falls es in den Einstellungen einen Eintrag zu JIRA gibt. Ist das nicht der Fall, wird nur der Status des Supportfalls geändert.

Grundsätzlich dient dieses Dialog-Fenster dazu die Informationen festzulegen, die das Ticket beim Erstellen erhalten soll. Dabei werden einige Informationen aus denen des Supportfalls abgeleitet, wie beispielsweise der Titel oder die Priorität (siehe Abschnitt 4.1.4). Mit diesen Informationen werden automatisch die Eingabefelder befüllt. Diese können jedoch vom Anwender jederzeit geändert werden. Darüber hinaus können zusätzliche Informationen angegeben werden, wie beispielsweise die Komponenten oder die Versionen, die von dem Ticket betroffen sind. Damit für diese Felder nur gültige Werte eingegeben werden können, werden die Metadaten zu dem Projekt, zu dem das Ticket hinzugefügt werden soll, von JIRA abgefragt (siehe Abbildung 3.9). Nachdem diese Metadaten aufbereitet wurden, werden dem Anwender zu den entsprechenden Feldern die möglichen Eingabemöglichkeiten zur Auswahl gestellt.

Sobald der Anwender alle Informationen eingegeben hat, die benötigt werden, kann das Ticket erstellt werden. Daraufhin wird eine entsprechende Anfrage an JIRA gesendet (siehe Abbildung 3.10), das Dialog-Fenster geschlossen und der Supportfall um die entsprechenden Ticket-Informationen ergänzt. Wird dagegen die Aktion abgebrochen, wird kein Ticket erstellt und der Supportfall entsprechend nicht der Entwicklung zugewiesen.

4.4. Anfragen an externe Webservices

Die App muss auf externe Webservices zugreifen, um die Kommunikation zwischen Zoho CRM und JIRA zu ermöglichen. Bei solchen externen Zugriffen muss beachtet werden, dass es aus Sicherheitsgründen das SOP (Same Origin Policy) Sicherheitskonzept gibt. Dieses untersagt clientseitigen Skriptsprachen wie JavaScript auf Objekte zuzugreifen, die von einer anderen Webseite stammen [17]. Durch die Unterbindung dieser domainübergreifenden Anfragen schützen sich Webanwendungen und Browser vor Angriffen. Dies hat maßgeblich Auswirkung auf die Entwicklung der App. Diese kann lediglich clientseitigen Code ausführen (siehe Abschnitt 3.3.3), muss allerdings auf die REST APIs von JIRA und Zoho CRM zugreifen können, um den Anforderungen (siehe Abschnitt 3.2) gerecht zu werden.

SharePoint stellt für solche Fälle den Webproxy bereit, der JavaScript trotz des SOP Sicherheitskonzepts Anfragen an externe Webseiten und -services ermöglicht. Dabei werden Anfragen nicht mehr direkt an die externen Seiten gestellt, sondern in eine Anfrage an den Webproxy verpackt, der anschließend die eigentliche Anfrage ausführt. Der Webproxy befindet sich in derselben Domain wie die App und führt die Anfragen bei sich serverseitig aus. Dadurch wird das SOP Sicherheitskonzept nicht verletzt und die Anfrage kann erfolgreich ausgeführt werden. Dieser Ablauf wird in Abbildung 4.1 veranschaulicht.

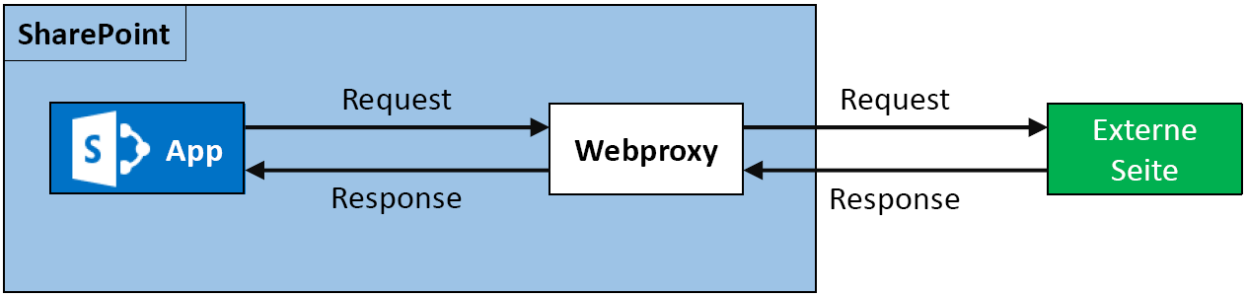

Abbildung 4.1.: Anfragen an externe Webservices über den Webproxy

Der Webproxy hat selbst auch ein Sicherheitskonzept und lässt aufgrund dessen nur Anfragen an bestimmte URLs zu. Damit eine Anfrage zugelassen wird, muss dem Webproxy die entsprechende URL bekannt gemacht werden. Dies wird durch einen Eintrag in die AppManifest.xml erreicht. Diese Datei enthält allgemeine Informationen über die App, wie beispielsweise welche Seite die Startseite ist oder welche Sprachen von der App unterstützt werden. Wird in der App eine Anfrage an eine URL gestellt, die nicht in dieser Datei eingetragen wurde, lehnt der Webproxy diese ab und liefert eine entsprechende Fehlermeldung zurück. Das Listing 4.5 zeigt einen beispielhaften Eintrag einer URL in der AppManifest.xml.

```
1 <RemoteEndpoints>
2     <RemoteEndpoint Url="http://www.beispiel.de" />
3     <!-- weitere URLs möglich -->
4 </RemoteEndpoints>
```

Listing 4.5: Eintrag der URL in die AppManifest.xml

Nachdem die URL in der AppManifest.xml hinterlegt wurde, werden entsprechende Anfragen vom Webproxy nicht mehr zurückgewiesen. Um nun eine Anfrage an die URL zu stellen, muss diese in eine Anfrage an den Webproxy verpackt werden. Das Listing 4.6 zeigt die dafür notwendigen Schritte.

Für Anfragen die über den Webproxy erfolgen sollen, stellt JSOM das *SP.WebRequestInfo* Objekt bereit. Diesem werden über verschiedene Setter-Methoden die Informationen zu der entsprechenden Anfrage übergeben. Das Objekt verpackt intern alle so erhaltenen Informationen in eine Anfrage an den Webproxy. Dieser benötigt zur Ausführung den aktuellen *ClientContext* und das *SP.WebRequestInfo* Objekt und erhält sie über die *invoke* Methode. Ausgeführt wird die Anfrage allerdings erst nachdem die *executeQueryAsync* Methode des *ClientContext* aufgerufen wurde.

Diese wartet nicht blockierend auf die Antwort und bekommt aus diesem Grund jeweils eine Methode übergeben, die im Erfolgs- oder Fehlerfall ausgeführt werden soll. Im Erfolgsfall hat der Webproxy die Anfrage erfolgreich ausgeführt und die erwartete Antwort zurückgeliefert.

```
1  var ctx = SP.ClientContext.get_current();
2
3  var request = new SP.WebRequestInfo();
4  request.set_url('http://www.beispiel.de');
5  request.set_method('GET');
6
7  var response = SP.WebProxy.invoke(ctx, request);
8
9  ctx.executeQueryAsync(function () {
10     // Request erfolgreich!
11 },
12 function () {
13     // Request fehlgeschlagen!
14 });
```

Listing 4.6: Anfrage über WebProxy an einen externen Webservice

4.5. Lokalisierung der App

Die App soll sowohl Deutsch als auch Englisch als Sprache unterstützen, damit sie für eine größere Zielgruppe interessant ist (siehe Abschnitt 3.2). Aus diesem Grund musste die App im Rahmen dieser Arbeit für beide Sprachen eigene Sprachdateien bereitstellen. Mit diesen soll abhängig davon, aus welchem Land die App aufgerufen wird, die entsprechende Sprache geladen und die App in dieser angezeigt werden. Kommt der Aufruf aus einem Land, zu der die App keine Sprachdatei anbietet, sollen die Ausgaben der App auf Englisch angezeigt werden.

Durch Anfragen teilen die Clients SharePoint mit, aus welchem Land sie kommen und in welcher Sprache die Inhalte angezeigt werden sollen. Die so angeforderten Sprachen können mit einem von SharePoint bereitgestellten Befehl zur Laufzeit ermittelt werden. Dieser wird im Rahmen dieser Arbeit genutzt, um dynamisch die benötigte Sprachdatei in Form von JavaScript-Dateien zu laden. Die dazu notwendigen Schritte sind im Listing 4.7 zu sehen.

```
1  <script type="text/javascript" src="../scripts/Resources.
2      <SharePoint:EncodedLiteral runat='server'
           text='<%$Resources:wss,language_value%>'
           EncodeMethod='HtmlEncode' />
3  .js"></script>
4  <script type="text/javascript"
     src="../Scripts/Resources.js"></script>
```

Listing 4.7: Laden der benötigten Sprachdatei

Falls es für eine Sprache keine passende Sprachdatei gibt, wird mit dem von SharePoint bereitgestellten Befehl auch keine Datei eingebunden. Aus diesem Grund muss zusätzlich ein weiteres Skript eingebunden werden, indem überprüft wird, ob eine Sprachdatei geladen wurde oder nicht. Damit die Überprüfung erfolgreich ausgeführt werden kann, wird in jeder Sprachdatei zu Beginn

ein Namensraum reserviert. In dem zusätzlich geladenen Skript kann somit einfach überprüft werden, ob der Namensraum bereits existiert. Falls nicht, wird die englische Sprachdatei eingebunden, da in diesem Fall eine Sprache angefordert wurde, die die App nicht unterstützt. Diese Vorgehensweise wird in Listing 4.8 veranschaulicht.

```
1 if (typeof (resources) == 'undefined' || resources == null) {
2     document.writeln('<script type="text/javascript"
          src="../Scripts/Resources.en.js"></script>');
3 }
```

Listing 4.8: Prüfen, ob eine Sprachdatei eingebunden wurde

Durch die in Listing 4.7 und 4.8 gezeigte Vorgehensweise wird sichergestellt, dass in jedem Fall eine Sprachdatei geladen wird. Diese JavaScript-Dateien enthalten alle Ausgaben in der entsprechenden Sprache in Form von einfachen Variablen, denen der entsprechende Text zugewiesen wurde. Diese Variablen müssen in beiden Sprachdateien identische Namen haben, damit auf den jeweiligen Seiten, unabhängig davon welche Sprachdatei geladen wurde, auf sie zugegriffen werden kann und somit der Text in der gewünschten Sprache angezeigt wird.

5. Ergebnisse

Das Ergebnis dieser Arbeit ist eine SharePoint App für den Datenabgleich zwischen Zoho CRM und JIRA. In diesem Abschnitt wird dieses Ergebnis anhand der Seiten der App präsentiert und deren Funktionsweise beschrieben.

5.1. Einstellungen

Damit die App mit Zoho CRM und JIRA kommunizieren kann, müssen zu diesen Anwendungen die URLs und die Anmeldeinformationen gespeichert werden. Für die Verwaltung dieser Informationen gibt es dazu die Seite Einstellungen. Diese stellt die momentan gespeicherten Informationen zu den Anwendungen als Tabelle dar, wie in Abbildung 5.1 veranschaulicht wird.

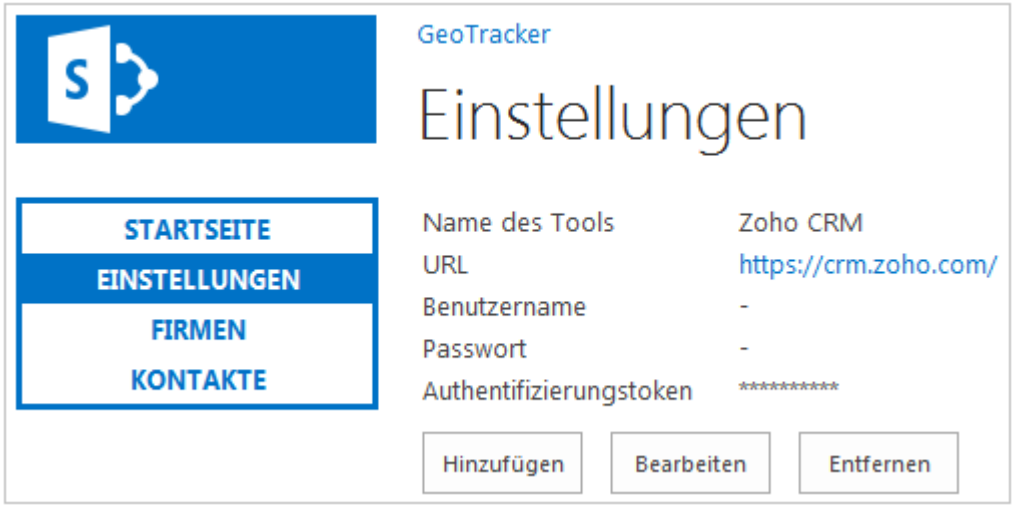

Abbildung 5.1.: Seite für die Verwaltung der Einstellungen

Durch die Buttons auf der Seite können zu den Einstellungen Informationen hinzugefügt, bearbeitet oder gelöscht werden. Dabei darf zu jeder der Anwendungen maximal einen Eintrag hinzugefügt werden. Das Bearbeiten und Löschen wiederum funktioniert nur, wenn mindestens zu einer Anwendung ein Eintrag vorhanden ist. Aus diesem Grund werden die Buttons entsprechend der Anzahl an Einträgen aktiviert oder deaktiviert.

Durch das Klicken auf einen der Buttons wird ein Dialog-Fenster geöffnet, das die gewünschte Operation ermöglicht. Die Abbildung 5.2 zeigt, wie dieses Dialog-Fenster für das Hinzufügen beziehungsweise Entfernen aussieht.

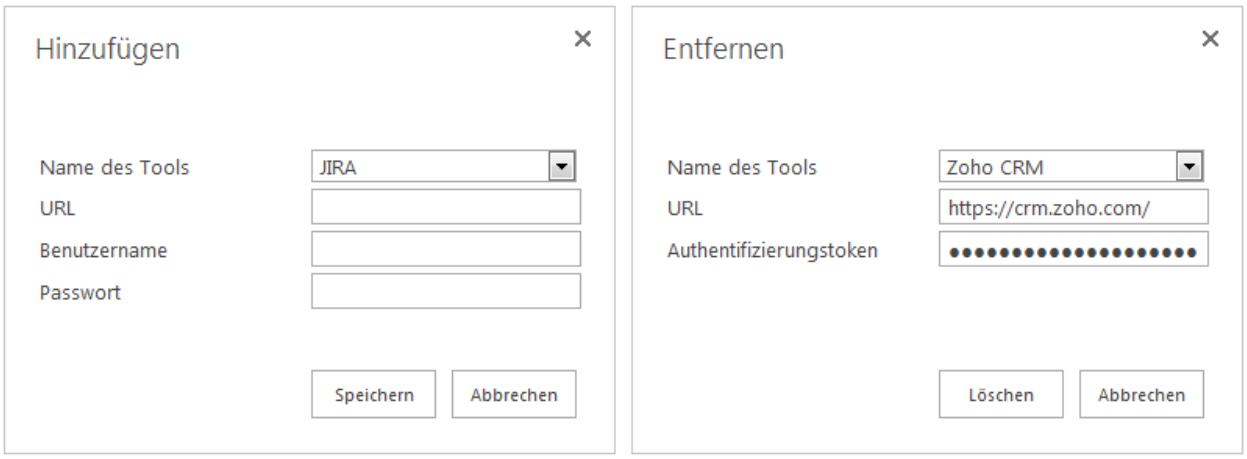

Abbildung 5.2.: Dialog-Fenster zum Hinzufügen und Entfernen von Einstellungen

Zu jeder Operation kann, wie in Abbildung 5.2 zu sehen ist, über eine Auswahlbox die Anwendung festgelegt werden, die von der Änderung betroffen ist. Um Fehler und Konflikte zu vermeiden, werden dabei beim Hinzufügen die Auswahlmöglichkeiten deaktiviert, zu denen es bereits einen Eintrag gibt. Bei den anderen beiden Operationen dagegen werden die Auswahlmöglichkeiten deaktiviert, zu denen noch kein Eintrag vorhanden ist, da nur vorhandene Einträge bearbeitet oder gelöscht werden können.

Wird der *Speichern* beziehungsweise *Löschen* Button gedrückt, wird geprüft, ob jedes Feld ausgefüllt wurde. Ist dies der Fall, wird das Dialog-Fenster geschlossen und die gewünschte Operation ausgeführt. Anschließend wird auf der Seite für die Einstellungen die Tabelle neu erstellt, um die angezeigten Informationen zu aktualisieren.

5.2. Supportfälle

Die Verwaltung der Supportfälle erfolgt auf der Startseite der App. Dort werden in einer Tabelle die momentan zur Bearbeitung offenen Supportfälle aufgelistet. Daneben befindet sich eine Auswahlbox, mit der die Supportfälle nach ihrem Status gefiltert werden können. Darüber hinaus befinden sich dort Buttons, mit denen der Datenaustausch mit Zoho CRM beziehungsweise JIRA gestartet werden kann. Diese Buttons sind nur sichtbar, wenn in den Einstellungen die entsprechenden Informationen zu den jeweiligen Anwendungen hinterlegt wurden.

Um die Supportfälle von Zoho CRM abzurufen, wird der Button *Zoho CRM Daten holen* gedrückt. Dadurch werden alle Supportfälle von Zoho CRM abgerufen, die nicht älter sind als der älteste Supportfall der App. Anschließend werden der App die Supportfälle, die noch nicht dort gespeichert sind, mit dem Status „Neu“ hinzugefügt. Ist ein Supportfall bereits vorhanden, werden dessen Informationen aktualisiert. Zum Abschluss wird die Tabelle, die die Supportfälle darstellt, neu erstellt, um die angezeigten Informationen ebenfalls zu aktualisieren. Wie die Tabelle nach

dem Abrufen der Supportfälle von Zoho CRM aussehen könnte, wird in Abbildung 5.3 beispielhaft veranschaulicht.

Nr.	Status	Titel	Projekt	Typ	Priorität	Fällig am
1	Neu	Test Supportfall 1	Sharepoint Portal	Aufgabe	Critical	2014-02-27
2	Kunde kontaktiert	Test Supportfall 2	Sharepoint Portal	Aufgabe	Minor	2014-02-27
3	Entwicklung zugewiesen	Test Supportfall 3	Sharepoint Portal	Bug	Blocker	2014-02-27
4	Ticket gelöst	Test Supportfall 4	Sharepoint Portal	Verbesserung	Major	2014-02-27
5	Erneut geöffnet	Test Supportfall 5	Sharepoint Portal	Aufgabe	Trivial	2014-02-27

Abbildung 5.3.: Tabelle der Supportfälle

In der Tabelle werden, wie in Abbildung 5.3 zu sehen ist, nur die wichtigsten Informationen angezeigt. Um eine detaillierte Ansicht aller Informationen eines Supportfalls zu erhalten, kann in der Tabelle auf einen Eintrag geklickt werden. Dadurch wird ein Dialog-Fenster geöffnet, das alle Informationen zu dem gewünschten Supportfall anzeigt. In Abbildung 5.4 wird beispielhaft ein solches Dialog-Fenster zu einem Supportfall gezeigt.

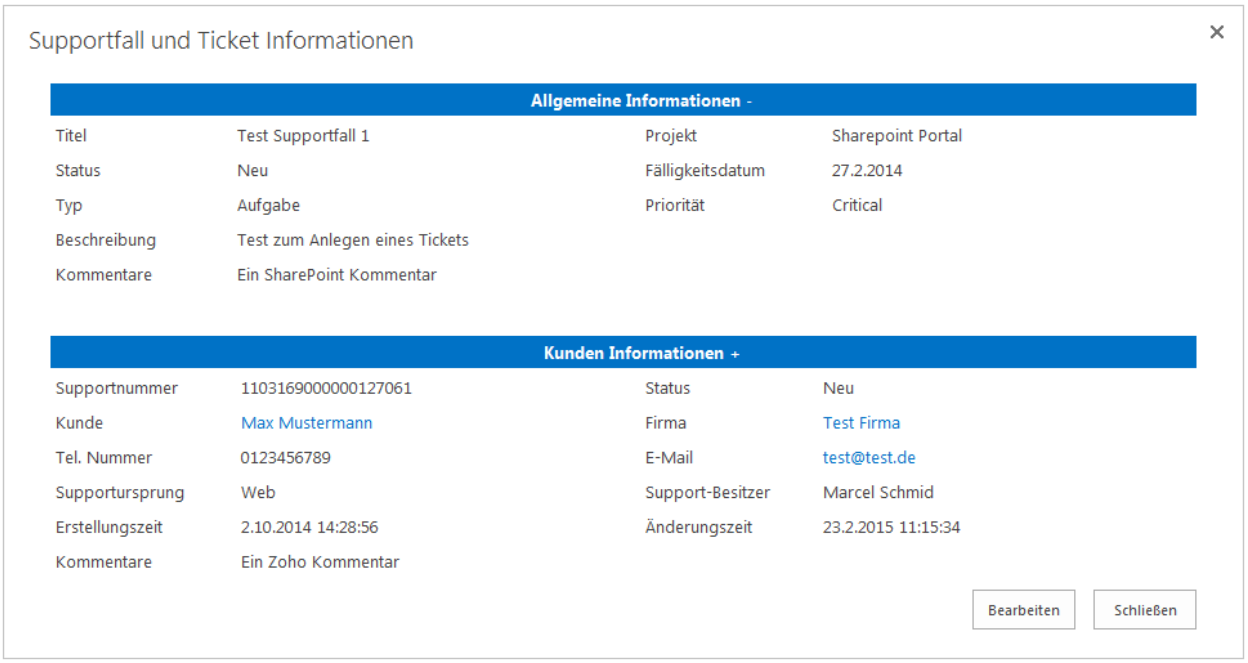

Abbildung 5.4.: Detaillierte Ansicht eines Supportfalls

Mithilfe dieses Dialog-Fenster ist es auch möglich, den Supportfall zu bearbeiten. Zu diesem Zweck werden Eingabefelder und Auswahlboxen eingeblendet, mit denen der Anwender nach Bedarf Änderungen vornehmen kann. Unter anderem ist es so möglich den Supportfall der Entwicklung zuzuweisen, indem der Status (siehe Abschnitt 3.4.3) entsprechend geändert wird. Falls in den Einstellungen ein Eintrag zu JIRA hinterlegt wurde, wird dafür ein weiteres Dialog-Fenster

geöffnet, das dem Anwender dabei hilft, ein Ticket in JIRA anzulegen. Die Abbildung 5.5 veranschaulicht, wie das Dialog-Fenster zum Erstellen eines JIRA Tickets aussieht.

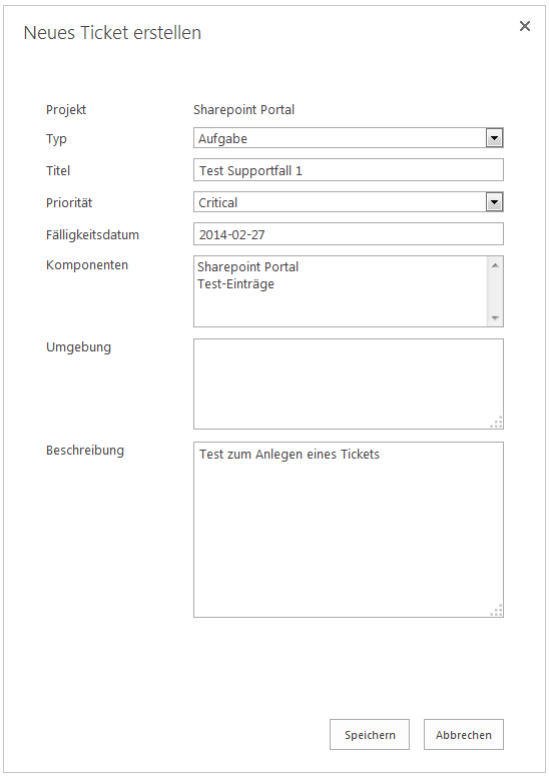

Abbildung 5.5.: Dialog-Fenster zum Erstellen eines JIRA Tickets

Mit dem in Abbildung 5.5 gezeigten Dialog-Fenster kann der Anwender die Informationen festlegen, die das JIRA Ticket erhalten soll. Wird anschließend der *Speichern* Button gedrückt, wird das Dialog-Fenster geschlossen und in JIRA ein neues Ticket angelegt. Des Weiteren werden die Ticket-Informationen und der Verweis auf das JIRA Ticket zum Supportfall gespeichert und in der detaillierten Ansicht des Supportfalls (siehe Abbildung 5.4) angezeigt.

5.3. Firmen und Kontakte

Neben den Supportfällen werden auch die Firmen und Kontakte von Zoho CRM in der App gespeichert und dargestellt. Auf diese wird jedoch nur lesend zugegriffen. Die Darstellung dieser Daten erfolgt ebenfalls mittels Tabellen, die nur die wichtigsten Informationen anzeigen. Für eine detaillierte Ansicht wird auch hier ein Dialog-Fenster geöffnet, sobald ein Eintrag in der Tabelle angeklickt wurde.

Im Gegensatz zu Supportfällen oder Kontakten zeigt das Dialog-Fenster zu einer Firma nicht nur sämtliche Informationen zu dieser an, sondern stellt auch alle Kontakte dar, die zu der Firma gehören. Diese Darstellung erfolgt wiederum als Tabelle, die nur die wichtigsten Informationen zu einem Kontakt anzeigt. Analog ist es auch hier möglich, durch Anklicken eines Eintrags ein

Dialog-Fenster zu öffnen, das sämtliche Informationen zu dem entsprechenden Kontakt darstellt. Die Abbildung 5.6 veranschaulicht beispielhaft, wie eine solche Auflistung aller Kontakte zu einer Firma aussieht.

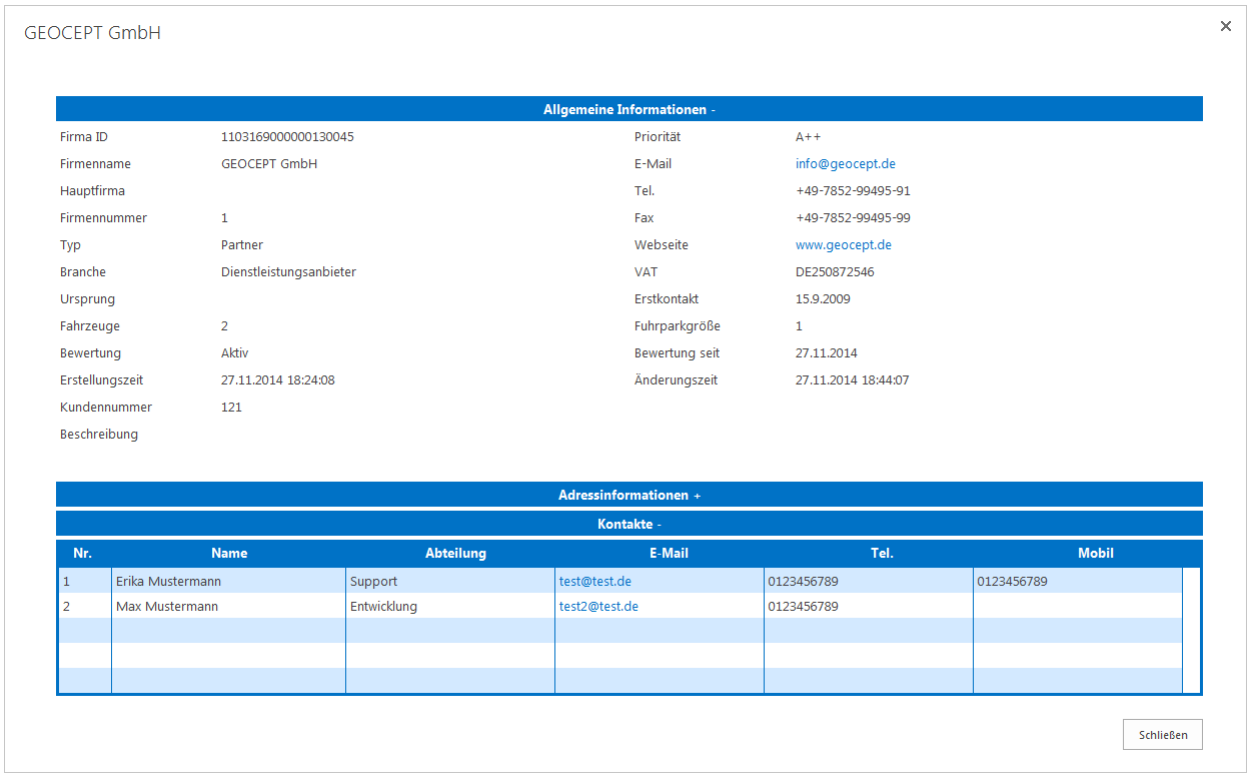

Abbildung 5.6.: Auflistung aller Kontakte zu einer Firma

6. Schlussbetrachtung

An dieser Stelle wird die Arbeit noch einmal auf das Wesentliche zusammengefasst und einen Ausblick auf mögliche Weiterentwicklungen gegeben.

6.1. Zusammenfassung

Im Rahmen dieser Arbeit wurde eine SharePoint App entwickelt, die den Datenabgleich zwischen Zoho CRM und JIRA regelt. Dazu musste zu Beginn das in SharePoint 2013 neu eingeführte App-Modell analysiert werden. Dieses Modell bietet verschiedene App-Varianten, die sich in der Art und Weiße unterscheiden, wie die Apps bereitgestellt werden. Die Varianten wurden evaluiert, wobei letztlich die Entscheidung auf die SharePoint gehosteten Apps fiel. Dies ist im Wesentlichen darauf zurückzuführen, dass für diese Apps kein zusätzlicher Applikationsserver bereitgestellt oder ein Hostingdienst in Anspruch genommen werden muss. Das hat jedoch zur Folge, dass es mit diesen Apps nicht möglich ist, serverseitigen Code auszuführen. Somit musste die gesamte Logik clientseitig mit JavaScript implementiert werden. Dadurch ergab sich unter anderem die Einschränkung, dass Anfragen an externe Webservices nicht direkt erfolgen dürfen, sondern nur über den von SharePoint bereitgestellten Webproxy möglich sind.

Neben der Evaluation der Hosting-Varianten musste auch ein Workflow für den Datenabgleich zwischen Zoho CRM und JIRA entworfen werden, der den Ablauf der Bearbeitung eines Supportfalls innerhalb der App beschreibt. Dabei musste betrachtet werden, in welchen Fällen eine Kommunikation mit einer der beiden Anwendungen stattfinden muss und welche Daten dabei ausgetauscht werden müssen. Als ersten Schritt musste die App dazu Supportfälle von Zoho CRM abrufen und speichern können. Diese können anschließend die durch den Workflow festgelegten Status durchlaufen, die den Bearbeitungsfortschritt eines Supportfalls darstellen. Unter anderem ist es dabei möglich, den Supportfall der Entwicklung zuzuweisen, wodurch mit den vorhandenen Informationen und den Eingaben des Anwenders ein Ticket in JIRA erstellt werden kann. Auch das Aktualisieren der Ticket-Informationen muss die App ermöglichen, wozu ebenfalls eine Kommunikation mit JIRA erforderlich ist. Wird ein Supportfall geschlossen, müssen die wichtigsten Informationen zurück nach Zoho CRM synchronisiert werden, damit dort die Supportfälle archiviert werden können.

Für die persistente Speicherung der Daten innerhalb der App gibt es in SharePoint lediglich die Möglichkeit SharePoint Listen zu verwenden. Diese entsprechen im Aufbau und der Funktions-

weise verallgemeinert Datenbanktabellen. Neben Supportfall- und Ticket-Informationen muss die App auch Einstellungen zu den Anwendungen, Firmen und Kontakte speichern können. Aus diesem Grund wurden mehrere Listen angelegt, die jeweils für die Speicherung der genannten Daten zuständig ist. Einzige Ausnahme sind die Supportfall- und Ticket-Informationen, die in eine Liste gespeichert werden, damit nur eine Liste und deren Elemente in den Kontext geladen werden müssen und für den Zugriff keine JOIN-Operationen notwendig sind. Dies hat jedoch zur Folge, dass viele Felder leer bleiben, falls zu einem Supportfall kein Ticket angelegt wird. Deshalb wurden die Supportfall-Felder von Zoho CRM an die Ticket-Felder von JIRA angepasst, sodass für Felder mit gleichem oder ähnlichem Inhalt nur eine Spalte in der Liste benötigt wird, um die entsprechende Information zu speichern.

Die gespeicherten Daten mussten für den Anwender aufbereitet und entsprechend dargestellt werden. SharePoint stellt für die Listen zu diesem Zweck eigene Seiten bereit, mit denen darüber hinaus auch Einträge der Liste hinzugefügt, bearbeitet oder gelöscht werden können. Diese konnten jedoch nicht verwendet werden. Der Anwender würde dadurch zu viele Rechte erhalten und die App hätte keine Kontrolle mehr über den Inhalt der Liste. Aus diesem Grund musste eine eigene Darstellung der Listen erstellt werden. Dafür wurde eine tabellarische Ansicht gewählt, da die Listen ähnlich aufgebaut sind. In den Tabellen werden nur die wichtigsten Informationen angezeigt, damit sie übersichtlich bleiben. Mithilfe von Dialog-Fenstern kann ein Listeneintrag detailliert dargestellt werden, nachdem auf einen Eintrag in der Tabelle geklickt wurde. Darüber hinaus wurden diese Dialog-Fenster verwendet um für die Einstellungen das Hinzufügen, Bearbeiten und Löschen von Einträgen zu ermöglichen oder aus einem Supportfall ein Ticket zu erstellen.

6.2. Ausblick

Die im Rahmen dieser Arbeit entwickelte SharePoint App wird zukünftig bei der GEOCEPT GmbH eingesetzt werden. Durch einen produktiven Einsatz können sich im Laufe der Zeit die Anforderungen an die App ändern oder neue hinzukommen. Aus diesem Grund sollte die App stets weiterentwickelt und gepflegt werden, damit sie immer den aktuellen Gegebenheiten gerecht wird. In den nachfolgenden Abschnitten werden deshalb mögliche erste Schritte bei der Weiterentwicklung der App genauer betrachtet und beschrieben.

6.2.1. Anlegen von Supportfällen, Firmen und Kontakten

Momentan fungiert die SharePoint App rein als Bindeglied zwischen Zoho CRM und JIRA. So müssen Supportfälle, Firmen und Kontakte in Zoho CRM angelegt und durch einen Knopfdruck in die App übertragen werden. Zukünftig wäre es hingegen durchaus wünschenswert, dass ein Supportmitarbeiter direkt in der App entsprechende Einträge anlegen kann. Dadurch müssten nicht Zoho CRM und die SharePoint App parallel bedient werden. Im Support könnte so ausschließlich

mit der SharePoint App gearbeitet werden, wodurch eine Trennung zwischen Support und Vertrieb gewährleisten werden könnte, da dort ausschließlich mit Zoho CRM gearbeitet wird.

6.2.2. E-Mail Benachrichtigungen

Treten in der App Änderungen an einem Supportfall auf, gibt es keine Möglichkeit andere Mitarbeiter darüber zu informieren. So bekommt die Entwicklung nur mit, dass ihnen ein Supportfall zugewiesen wurde, wenn sie in JIRA oder in der App nachsieht. Andersherum bekommt der Support nicht mit, dass ein Ticket von der Entwicklung behoben wurde, ohne in der App nachzusehen. Um dies zu ändern, könnten E-Mail-Adressen für die Entwicklung und den Support in der App gespeichert werden. Sobald ein Supportfall der Entwicklung zugewiesen wurde oder diese ein Ticket behoben hat, könnten automatisch E-Mail Benachrichtigungen an die jeweilige Abteilung erfolgen. Dadurch bekommen alle beteiligten Mitarbeiter mitgeteilt, dass es Änderungen gab, auch wenn diese die App nicht gestartet haben.

Literaturverzeichnis

[1] Wikipedia - Die freie Enzyklopädie. (Nov. 2014). Telematik, Adresse: http://de.wikipedia.org/wiki/Telematik (besucht am 20. 11. 2014).

[2] GEOCEPT GmbH. (Nov. 2014). GEOCEPT GmbH: Überblick, Adresse: http://www.geocept.de (besucht am 20. 11. 2014).

[3] Wikipedia - Die freie Enzyklopädie. (Dez. 2014). SharePoint, Adresse: http://de.wikipedia.org/wiki/SharePoint (besucht am 27. 12. 2014).

[4] S. Wright, *Pro SharePoint 2013 App Development*, Ser. Expert's voice in SharePoint. Apress, 2013, ISBN: 978-1-4302-5885-8.

[5] S. Kostojohn, B. Paulen und M. Johnson, *CRM Fundamentals*, Ser. Apresspod Series. Apress, 2011, ISBN: 978-1-4302-3590-3.

[6] Wikipedia - Die freie Enzyklopädie. (Dez. 2014). Jira (Software), Adresse: http://de.wikipedia.org/wiki/Jira_(Software) (besucht am 26. 12. 2014).

[7] M. Gelbmann. (Aug. 2012). jQuery now runs on every second website, Adresse: http://w3techs.com/blog/entry/jquery_now_runs_on_every_second_website (besucht am 15. 12. 2014).

[8] R. Steyer, *jQuery: das JavaScript-Framework für interaktives Design*, Ser. Open source library. Pearson Deutschland GmbH, 2011, ISBN: 978-3-8273-3072-7.

[9] Wikipedia - Die freie Enzyklopädie. (Sep. 2014). jQuery, Adresse: http://de.wikipedia.org/wiki/JQuery (besucht am 15. 12. 2014).

[10] R. T. Fielding. (2000). Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST), Adresse: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (besucht am 23. 12. 2014).

[11] L. Richardson und S. Ruby, *RESTful Web Services*. O'Reilly Media, 2008, ISBN: 978-0-596-55460-6.

[12] Wikipedia, the free encyclopedia. (Dez. 2014). Representational state transfer, Adresse: https://en.wikipedia.org/wiki/Representational_state_transfer (besucht am 23. 12. 2014).

[13] JSON Group. (Dez. 2014). Introducing JSON, Adresse: http://www.json.org/ (besucht am 14. 12. 2014).

[14] Wikipedia - Die freie Enzyklopädie. (Juli 2014). Extensible Markup Language, Adresse: https://de.wikipedia.org/wiki/Extensible_Markup_Language (besucht am 25. 12. 2014).

[15] Zapier. (2014). Jira & Zoho CRM Integrations, Adresse: https://zapier.com/zapbook/jira/zoho-crm/ (besucht am 29. 12. 2014).

[16] Microsoft Developer Network (MSDN). (7. Okt. 2014). Übersicht über Apps für SharePoint, Adresse: http://msdn.microsoft.com/de-de/library/office/fp179930(v=office.15).aspx (besucht am 02. 01. 2014).

[17] Wikipedia - Die freie Enzyklopädie. (Nov. 2014). Same-Origin-Policy, Adresse: http://de.wikipedia.org/wiki/Same-Origin-Policy (besucht am 13. 01. 2014).

A. Anhang

Tabelle A.1.: Alle Anpassungen der Zoho CRM Felder für Supportfälle

SharePoint Spalten	JIRA Felder	Zoho CRM Felder	
SharePoint Status	-	-	
Supportnummer	-	Support-Nummer	
Ticket Schlüssel	Ticket Schlüssel	-	
Projekt	Projekt	Produkt-Name	gleicher Inhalt
Typ	Vorgangstyp	Typ	Auswahl angepasst
Titel	Zusammenfassung	Betreff	gleicher Inhalt
Priorität	Priorität	Priorität	Auswahl angepasst
Fälligkeitsdatum	Fälligkeitsdatum		neues Textfeld
Komponenten	Komponenten		neues Textfeld
Betrifft Version(en)	betrifft Version		neues Textfeld
Betrifft Revision(en)	betrifft Revision	-	
Lösungsversion(en)	Lösungsversion		neues Textfeld
Lösungsrevision(en)	Lösungsrevision	-	
Bearbeiter	Bearbeiter	-	
Umgebung	Umgebung		neues Textfeld
Beschreibung	Beschreibung	Beschreibung	gleicher Inhalt
Verbleibende Schätzung	Verbleibende Schätzung	-	
JIRA Kommentare	Kommentare		neues Textfeld
Zoho Kommentare	-	Kommentare	
SharePoint Kommentare	-		neues Textfeld
JIRA Status	Status	-	
Zoho Status	-	Status	„Erneut geöffnet“ hinzugefügt
JIRA Änderungszeit	Aktualisiert	-	
Zoho Änderungszeit	-	Änderungszeit	
Status der Lösung	Lösung	Lösung	gleicher Inhalt

Supportursprung	-	Supportursprung	
Kunde	-	Verknüpft mit	
Firma	-	Firma-Name	
Tel. Nummer	-	Tel.	
Supportbesitzer	-	Support-Besitzer	
E-Mail	-	E-Mail	
Zoho Erstellungszeit	-	Erstellungszeit	